47－R004

システム評価方式の調査

昭和48年3月

JIPDEC

財団法人　日本情報処理開発センター

この報告書は，日本自転車振興会から競輪収益の一部である機械工業振興資金の補助を受けて昭和47年度に実施した「システム評価方式の調査」の成果をとりまとめたものであります。

序

最近，わが国におけるコンピュータの利用は，情報化社会の進展とも相まって，ますます拡大し高度化される方向にあります。しかし一方では，設備および運用に，莫大な費用を必要とするこの機械に対し，果してそのコストに見合う効果を発揮しているか否かについてさらに検討を要する時期であるとも言われております。

ハードウェアとソフトウェアを包含するコンピュータ・システムの評価に関する研究は米国においては 1960 年代に入って本格化し，わが国においても近年コンピュータの製造者，利用者を問わず，広い関心を集める大きなテーマの一つとなってまいりました。

当財団では国の内外におけるこれらの評価技術の基本的性質，およびそれらの技法を実際に適用するにあたって発生する問題点などに関し，広く調査を進めてまいりましたが，本報告書はその成果をまとめたものであります。

本報告書が各方面に利用され，わが国情報処理技術向上の一助として寄与できることを念願いたす次第であります。

財団法人　日本情報処理開発センター
会長　難波捷吾

目　　次

まえがき

本報告書は，「コンピュータ・システムの評価方式」に関する，内外の各種文献および資料を中心に調査をおこない，その成果をまとめたものである。

近年におけるコンピュータ技術の発展，およびその適用分野の拡大には目覚しいものがある。

反面，これらの技術はあまりにも急速に進展したため，コンピュータそのもの，あるいはそれを含むシステムの技術的評価の側面は，コンピュータの発展に比べ，著しく遅れていたと言わざるを得ない。

コンピュータに関する技術（Computer technology）が，いわゆるコンピュータ・サイエンス（Computer Science）として科学の一分野となり得るためには，評価技術も含めたコンピュータ・テクノロジーを体系化し，より一層の発展を図る事が急務となる。

このような事情から，最近になってこれらの評価技術は，コンピュータ・システムの個々の分野で部分的にとりあげられ，徐々にその成果を表わしつつあり，その重要性がますます認識されてきている。

本報告書は，全六章から構成され，各章はそれぞれ以下のような内容についてまとめられている。

第1章では，コンピュータ・システム評価の背景およびこれについて現在まで行われた研究の変遷のようすを探り，次章以降への導入部となっている。

第2章は，コンピュータ・システムの評価に関する技術を体系的にとらえ，その技法の各々についての詳細を解説し，実施例を中心にその分野での研究の概要を述べると共に，今後に残された問題点を整理したものである。

第3章では，コンピュータ・システムの評価技術のうち，商用システムとして利用者の便に供されている各種のシステムについて，その種類・機能を紹介し，代表的なものについてやや詳細な解説をおこなっている。

第4章は，主にタイムシェアリング・システムなど大型コンピュータ・システムの評価における問題点，研究の方向を概観した後，現実にどのようなアプローチがなされているかをケース・スタディと共に解説したものである。

第5章は，近年特に脚光を浴びているコンピュータ・ネットワークについて，その概況を解説し，このような大型のネットワーク・システムにおける評価の技術および問題点についての現況を整理し解説したものである。

第5章は，米，英，仏3ケ国におけるシステム評価の現状の比較，および，商用評価システムのユーザの使用経験などを中心に海外調査をおこなった結果をとりまとめたものである。こゝでは要約にとゞめ，より細部にわたっての記述は別冊"日本情報処理開発センター　資料

コンピュータ・システム評価　海外調査報告書"

にゆづる。

1．コンピュータ・システム評価の概要

1. コンピュータ・システム評価の概要

1.1 コンピュータ・システム評価の背景

英国の著名な作家Oscar Wildeは言った。

"A man who knows the price of everything, and the value of nothing"

彼の死後70年，今世紀の後半に入ってのこの異様なコンピュータ社会の真只中で，人々は彼の言葉の真実を身をもって体験した。

1972年末現在，米国におけるコンピュータの設置台数は6万を越え，我が国のそれも15000台を記録した。

コンピュータはたしかに文明の利器ではある。しかし一部では"金食い虫"あるいは"紙屑製造機"などの汚名すら着せられ，それ自体が高価であるばかりでなく，更に多額の設備費や運転費を食い，加えて予想に反した漠大な人件費を必要とするこの稀有な機械が，本当にそのコストに見合う効果を発揮しているのかどうかが疑問視されるのは当然すぎる程当然であろう。

コンピュータ・システムの評価とは，一応3つの立場にわけて考えられる。

第1は，テクノロジー・アセスメント（Technology Assessment）と類似のフィロソフィによるコンピュータ・アセスメント（Computer Assessment）という立場である。即ち，コンピュータそのものの存在あるいは利用自体を評価するものである。このところ"コンピュータ公害"，"情報公害"などの声が高まり，社会的問題として批判され始めている。

第2は，よりプラクティカルな立場からの，主として経済的あるいは経営的な観点からみたコンピュータの価値論である。世間の流行に遅れまいと，無批判にコンピュータを導入した為，かえって経営上の支障を来たしたと思われる企業も少なくはないが，そのコストに見合った効果に対する再検討は，ここ数年来，多くのコンピュータ・ユーザの大きな関心事である。例えばもしその仕

事を人手で行なったとしたら，どれだけの費用がかかるであろうかという置き換えによる比較をはじめ，コンピュータの効果を定量化する努力がかなり行われて来ている。

第3は，コンピュータそのものの利用は肯定した上で，より効率よく利用する，あるいはよりよいコンピュータを利用するための評価という立場である。

この報告書では主としてこの第3の立場から，特に技術的な問題を中心に扱うこととする。

1.2 評 価 の 項 目

コンピュータ・システムの評価は，ハードウェア，ソフトウェアおよびその綜合システムが対象となる。

ハードウェアの評価項目としては，

① 処理方式（先行制御，平行処理など）

② CPUの処理速度（メモリーサイクル，命令実行時間）

③ インストラクションの種類と機能

④ 信頼性（MTBF，平均故障間隔）

⑤ 保守性（MTTR，修理に要する平均時間）

⑥ 稼動率（保守，故障のための時間を除いた稼動可能時間の率）

などがある。

最近のコンピュータではソフトウェアの影響もあり①，②，③の生の機能はそのまゝ直接にはユーザに反映しなくなっている。

次にソフトウェアの評価項目としては，

① 機能（仕様上の機能）

② 処理速度

③ リソース占有量（メモリ占有量）

④ 精度（計算の精度）

⑤ 操作性（使い易さ）

⑥ 拡張性（機能の拡張）

⑦ 信頼性

等があるが，これはベーシック・ソフトウェアかアプリケーション・ソフトウェアかによっても，またアプリケーションの種類によってもそれぞれやゝ異なる要素を持つ。最近は，アプリケーションの種別毎に何等かの評価基準を設定するという動きも出て来ている。

一方，制御プログラムあるいは言語プロセッサ等のベーシックなものは，ソフトウェアのみをとり出して評価すること自体にやゝ問題があり，むしろハードウェアと綜合して考慮すべき点が多い。特に制御プログラムに関してはその傾向が強い。

綜合システムの評価項目としては

① 処理効率

② オーバヘッド（ overhead ）

③ 各リソースの利用率およびそのバランス

④ スループット（ throughput ）

⑤ ターンアラウンド・タイム（ turn around time ）

⑥ レスポンス・タイム（ response time ）

⑦ 操作性

等があげられる。

マルチ・プログラム，マルチ・プロセッシングなどの処理形態や，キャシュ・メモリー（ cash memory or buffer memory ）の付加，あるいはバーチャル・メモリー（ virtual memory ）の概念などの導入によって，システムの動きはハードウェア，ソフトウェアがからんできわめて複雑なものとなって来た。

①と②はかなり関連の深い項目であり，①はある処理（主としてユーザ・ジョブ）の開始から終了までの時間内における，そのジョブ自体の走行時間の比率である。即ち，制御プログラムの走行時間や，アイドル・タイム（ idle time ）の占める割合と，実質的なユーザプログラム処理の占める割合から効率の目安が得られる。②のオーバヘッドは通常，ユーザプログラムと制御プログラムの処理時間の比をいう。また基本単位の処理を行う場合にも最低限必要となる制御プログラムの処理量のことをいう場合もある。例えば，１つのジョブの開設のためのイニシャライズ，終結のためのターミネート処理，各種制御テーブルへの登録処理，マルチ処理のためのスワッピング（ swapping ）やスケジューリング（ scheduling ）等の基本的な機能が一切含まれる。タイムシェ

アリング・システムにおいては，オーバヘッドが７０～８０％という様なこともかなり現実に起っている。またメモリー・オーバヘットとしてメインメモリーのレジデント（ resident ── 制御プログラムが居すわっているエリア ）容量を問題にする場合がある。

③はハードウェア構成の各リソースがどの程度効率よく利用されているか，また夫々がうまくバランスよく利用されているかということで， CPU や各チャネルあるいはメモリーなどが夫々平均的に利用されていることが望ましいが，例えば特別なチャネルが何時もビジイ（ busy ）であって他は遊んでいる，あるいはメモリー容量の制約に押さえられ内容の入れ変えのみに忙しく CPU が充分利用されていない等のアンバランスは処理効率を落すばかりでなく，経済的にも無駄があることとなる。

④および⑤は①②③などと無関係ではないが，むしろシステム運用上の評価の目安である。スループットはシステム全体としての単位時間内の処理量で，マルチ処理によって向上させることが出来る。一方ターンアラウンド・タイムは個々のジョブの処理時間であり，スループットの向上とターンアラウンドの向上は必ずしも同期しない。

⑥のレスポンス・タイムは主としてオンラインあるいはタイムシェアリング・システムにおいて問題となる。最近のシステムは，少くとも３秒以内のレスポンス・タイムが要求されている。

⑦の操作性はバッチ処理の場合は，主としてオペレータの操作のし易さが中心であったが，人間と機械の間のインタラクション（ interaction ）が極めて強くなった最近のマン・マシーン・システム（ man machine system ）では，単に使い易いと言うだけでなくむしろ人間工学的な，あるいは心理学的な角度からの評価を必要として来ている。

以上述べたようにシステム評価は，ハードウェア，ソフトウェア，そしてその綜合システムが対象となるが，この報告書では特に最後の綜合システムに対する評価を中心として考える。但しこゝで一言つけ加えておかねばならぬことは，最近の傾向はハードウェアとソフトウェアとソフトウェアのみではなく更

にそれを使用する人間即ちヒューマンウェアを含めて1つのシステムと考える傾向が出て来たことである。今日のようにオンライン利用が日常化してくると，オンライン端末におけるオペレータ，タイムシェアリング・システムで端末からシステムを使用するユーザなどは当然システム系の一部と考えられる。そこで端末における人間の振舞い（behavior）を分析する研究がかなり行われ始めている。このようにしてシステム評価は益々複雑な要素を含んで来ることとなった。

1.3 システム評価の変遷

コンピュータ・システム評価に関する論文は，すでに 1940 年代に BTL (Bell Telephone Laboratory)の Shannon 等によって，発表されたと言われている。たゞし 1950 年代までの評価の対象は専らハードウェアが中心であり，CPU の処理速度および安定性などが主として注目された。しかもこれらの概念や技術の多くは国防や宇宙開発のために開発されたものであった。

ソフトウェアも対象として考えられるようになったのは 1960 年以降のことであり，システム選択のための評価が本格化したのは 1960 年代の中頃であった。評価の目標もスループットに対するコスト効率が強調されるようになった。これらの活動は矢張り NASA や軍関係の機関が最も強力な推進者であり，その委託研究の成果も含めて，この頃から数多くの研究実績が発表されるようになった。

IBMユーザの団体である SHAREの中にコンピュータ・システム評価に関する委員会が作られたり，コンピュータ・サイエンスと統計学のインタフェース (interface) として " Compumetrics " という言葉が生れたりした。

1960 年代の後半から 1970 年代にかけて各種のコンピュータ・コンファレンスにおいてもシステム評価の問題がかなり取り上げられるようになり，広い興味と関心を呼ぶ大きなテーマの一つとなって来た。

参考までに米国における主要コンファレンスである SJCC，FJCC，ACM National Conference，および Conferences on Application of Simulation 等における 1945 年から 1970 年迄の発表論文のうち，システム評価に関するものの件数を年度別に示す。

・ 1945−1
・ 1948−1
・ 1950−1
・ 1953−1
・ 1960−7
・ 1961−6
・ 1962−9
・ 1963−7
・ 1964−14
・ 1965−8
・ 1966−13
・ 1967−23
・ 1968−31
・ 1969−62
・ 1970−50

註：この表は

Measurement of Computer Systems −An introduction

A. F. Goodman,

Mcdonnell Douglas Astronautics Company

FJCC 1972 より引用した。

1971年以後はSJCC，FJCC，ACM annual conference の37のcroceedingsを見た限りでは1971年11件，1972年39件と，一見減少しているかのようにみえるが，この頃から，システム評価のspecial interest groupや分科会的活動がさかんとなり，それらの資料を含めれば実質的にはより多くなっている。

コンピューのシステム評価が，具体的に必要になる主な場面としては

① 新規の導入，あるいは入れ換えの場合の機種選定

② 現在のシステムの性能や効率の改善

③ 新しいシステムの開発，設計時の効率の予測

などがある。

このところ政府機関を含め，米国におけるコンピュータ・ユーザの間では，上記のような各場面で，システムを測定し，評価するプロセスはかなり定着して来ている。政府機関の導入機種の選択は，定量的な効率予測の評価プロセスが半ば義務付けられており，また民間にあっても比較的大きなユーザは評価のための人手，コンピュータ時間あるいは費用などはさほど惜しまずに消費して

いるようである。

このような背景の下で，米国では商用の評価システムの普及はめざましい。商用システムには

① シミュレータ

② ハードウェア・モニター

③ ソフトウェア・モニター

の3種類があるが，特にIBMの機種を対象としたものを中心に数百のオーダのユーザを持つパッケージもある（これら商用システムに関しては第3章で紹介している）。

このような米国の事情に対して，我が国やヨーロッパ諸国では，全般的にかなり遅れている（第6章参照）。我が国では数年前から，大学や研究機関の間でシミュレーションを中心とした研究が行われて来たが，最近は全般的に認識が高まり，コンピュータ・メーカがハードウェア・モニターやソフトウェア・モニターなどの実用経験を重ね，更に比較的進んだユーザはベンチマークやシンセティック・プログラムあるいはモニタリングの経験を持つところも増えて来た。このような盛り上がりの成果として，1972年夏 情報処理学会の主催でシステム評価のシンポジウムが開かれたが，ほゞ時を同じくして英国でもBritish Computer Societyによって同様のコンファレンスが開催されている。

2. 評　価　手　法

2. 評　価　手　法

2.1 システム評価へのアプローチ

この節では，システム評価に関する今までのアプローチを整理し，システム評価に関する技術を体系的にとらえることにより，次節で述べる各手法がシステム評価に於てどう位置ずけされるかを明らかにする。また，それに関連して重要なwork load の問題に関して簡単な考察を加えておく。

計算機の発展の初期の段階に於ては，その処理能力は主にハードウェアの性能によって大きく左右されていた。この頃は，計算機システムを評価する上での関心は，専らハードウェアの性能に向けられていた。

即ち，CPU の処理速度，記憶容量，マシンの稼働率，信頼性，保守性，等がその中心であった。

特に，CPU の速度は，計算機の処理速度を代表する一つの目安として考えられ，CPU の性能を評価する方法がいくつか考案された。

まず最も単純な方法として，その当時，時間のかかる命令の代表として考えられていた。加算命令の実行時間と，メモリのサイクルタイムによってその性能を比較しようとする，Add & Cycle method が考え出された。

しかし，実際のプログラムに於て実行される命令群を，加算命令だけで代表すると云うのは，少し乱暴なやり方であり，もっと多くの命令を考慮する必要がある。この様な観点から，実際にいくつかのプログラム統計から，よく使用される命令についてその使用頻度を求め，それ等の重み付き平均として平均命令実行時間を求める方法が考案された。

これはインストラクション・ミックス（Instruction Mix）と呼ばれ，現在もなお，ハードウェア性能の指標の一つとして広く用いられている。

また，より広範囲のハードウェア特性を含めて評価する方法としてカーネル法（Kernel method）が考案された。これは，実際に標準的なプログラムをその機械のマシン語でコーディングし，マニュアル・データをもとにしてそ

の実行に要する時間を見積る方法である。

この方法は，前二者と比較して，より広範囲の命令を利用することになるし，またアドレッシング方式，インデック・レジスタ等，その他のハード的特性も考慮に入れられると云う点で優れていることになる。

しかし，いずれの方法も，ハードウェア性能そのものの評価基準としても，まだ十分なものとはなっていない。例えば，インストラクション・ミックス法について云えば，先まわり制御が行われている様な系では，その能力を正当に評価するには適当とは云えないし，又メモリのサイクル・タイムについてもメモリのインタリービング等について考慮しなくては，比較は余り意味が無いことになる。

その他，最近の計算機システムの様に，パイプ・ライン方式（Pipe Line），パラレル・プロセシング（Parallel Processing），バッファー・メモリ（Buffer Memory），マイクロ・プログラミング（Micro Programming）等の新しい技術がとり入れられるようになると，これらの単純な評価法による比較は成り立たず，その性能評価はますます困難になってきている。

この様に，ハードウェアの性能評価に於ても，各々の固有なハードウェア特性に機器構成を含めて綜合的にこれを評価すると云う事が重要になってきている。

一方，最近の計算機システムの様に，その処理にオペレーティング・システムが大きく介入してくるようになると，そのパフォーマンスは，ソフトウェアの性能によって大きく左右されるようになってくる。

即ち，ハードウェア性能と云うのはシステムの能力の一つの上限を与えるもので，そこにいかにして到達するかは，そのソフトウェアの質に大きく依存することになる。

この様な事情から最近では，ハードウェア，ソフトウェアを含めた総合システムとして，システムを評価しようとする方向に目が向けられるようになってきた。

一般に，システムを評価する時，その分析の仕方には，２つの立場が考えら

れる。

1つは現実のシステムを観測し，そこからパフォーマンスに関する情報を集めて，システムの振舞いを分析しようとする方法である。

もう1つは実際のシステムと等価な働きをするモデルを作成し，モデルを使ってシステムの振舞いを分析しようとする方法である。

前者はさらに，システムをどう扱うかと云う観点から，一般に次の2つのアプローチに分けて議論されている。

一方はStimulus approach と呼ばれるものであり，もう一方はAnalytic approachと呼ばれるものである。Stimulus approach では，システムの内部構造は見ずに，システムをブラック・ボックスとして扱おうとする。即ち，刺激（Stimulus）として，システムに負荷を与えてそれに対するシステムの応答（Response）を求め，その関係を分析することによってシステムの性格を調べようとするものである（図2－1）。

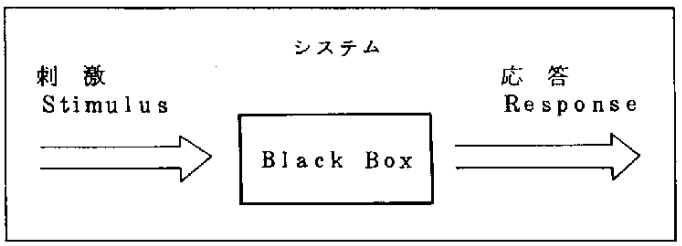


図2－1

この時，刺激として与えられる負荷は，人為的に作成される（Artificial work load）場合もあるし又，実際の負荷（Natural work load）がそのまま利用されることもある。

刺激の与え方，つまりどう云う性格の負荷を与えるかと云うことは，評価する側からその立場に応じて適切に決めてやる必要がある。

ベンチマーク・プログラム（Bench mark program）もシンセティック・プログラム（Synthetic program）も共に刺激として与えられるテスト・プログラムである。

前者が，プログラムとして固定された性格を持つの対し後者は，パラメータ化してその性格に柔軟性を持たせたと云う点が異る。この様に考えるとベンチマーク・プログラムもシンセティック・プログラムも云わば Artificial work load の作成技術であると云う見方もできる。

もう一方のアナリティック・アプローチ（Analytic approach）は，システム内部の状態をもっと深く観察することにより，システムの振舞いを分析しようとする行き方である。

これにはシステムの動作に関するデータを，もっと細部にわたって収集する必要がおきてくる。

この面での技術をモニタリングと呼んでいる。

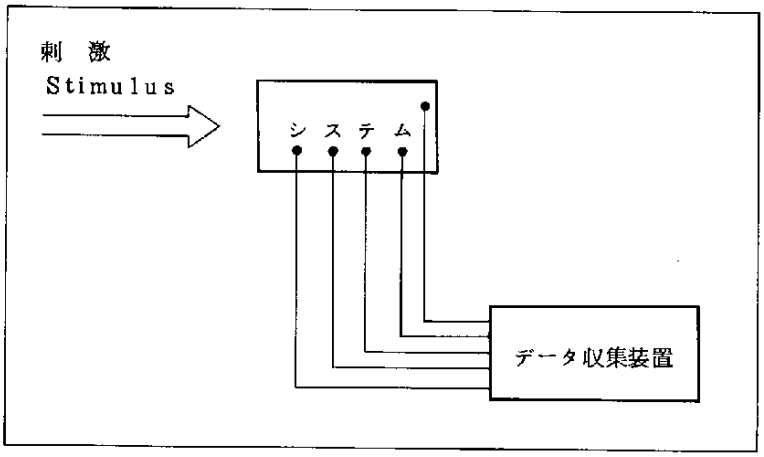


図2－2

モニタリングには，2つの方式がある。

一つはシステムから直接，電気的信号を取り出しこれを収集する方法で，これはハードウェア・モニタリングと呼ばれている。

もう一つは監視用のソフトウェアによってシステム内部の動作に関する情報を収集する方法である。これはソフトウェア・モニタリングと呼ばれている。モニタリングでは，システムの振舞いを把握するためには，どう云う性格のワーク・ロードを与え，どう云う量を測定対象として選ぶかと云う事，又得られた

データをどの様に処理してシステム評価と結びつけてゆくかと云う点が重要な問題となってくる。

一般に測定対象とされているものとしてはリソース・ユティリゼーションがあげられるが，これも評価する側の立場によってさまざまであり，何を測定すればよいかということは一概には云えない。どう云う量がシステムの振舞いに関してどの様な意味を持つかと云う点に関しては，まだ一般化された解答が得られていないのが現状であり，それを追求すること自体が一つのテーマにもなっている。得られたデータの整理に関しては，従来の統計的手法を適用するとか，モデルに取り入れて解析すると云った方法が試みられている。

モデルによる解析は，その扱うモデルの性格の違いから，シミュレーションと解析的手法の２つの手法に分類される。

シミュレーションは，対象とする系と等価な動作をするモデルを計算機内に作成し，これを用いてシステムの振舞いを解析しようとする方法である。

一般にその動作原理からEvent Oriented Simulation（事象追跡型シミュレーション）と呼ばれている。

シミュレーションは，モデルに比較的柔軟性があること，又かなり精密なモデル表現が可能であることなどからシステムの動作解析，システムの設計等に於て最も有力な武器とされている。

しかし一方では，実行時間の問題，モデルの妥当性のチェックが困難と云った問題が指摘されている。

解析モデルは計算機システムを数式モデルとして表現し解析する方法である。

多くは待ち行列理論に基くもので，計算機システムを窓口システムにたとえて解析する形をとる。現在の待ち行列理論では解析可能な範囲に限度があるため，余り複雑なシステムの解析には適さず主にシステム構成要素の一つに注目した，局所化された問題を扱うのに利用されている。しかし，最近は待ち行列理論の発展した形や，グラフ理論等の応用も試みられており，一部に解析モデルの限界を唱える説がなきにしもあらずであるが，むしろ研究そのものは活発になって来ている。すぐれた解析モデルの存在は，多くの努力を必要とする評

価のプロセスをかなり容易なものにするであろう。

以上，システム・パフォーマンスを評価するための２つのアプローチ，即ち実際のシステムを解析する方法とモデルによって解析する方法の２つのアプローチに関して，それに関連する技術を体系的に述べてきた。

これらの評価技術は，それぞれが独立に存在するだけでなく，お互いに他を助け，補い合ってゆく性質のものである。例えば，モニタリングの結果はシミュレーション・モデルの妥当性のチェックにきわめて有効である。またシミュレーションは解析モデルのパラメータの研究に有力な役割を果たす。

なおこれらの手法が，それぞれの評価目的，即ち機種の選定，既存システムの性能の改善および新システム設計時の予測等の各場面にどのように適用されるか，またそれぞれの利用の容易さ，精度，経済性等を表２－１に示す。

＊表２－１　評価方法の適用性

評価の方法		評価目的			利用の容易さ	経済性	精度
		機種の選定	性能の改善	新システムの性能予測			
Add & Cycle time		○	×	×	5	5	5
Instruction Mix		○	×	×	5	5	5
テストプログラム	Kernel	○	×	×	4	4	4
	Benchmark	○	○	×	4	3	3
	Synthetic Prog	○	○	×	3	3	3
解析モデル	Analytic model	○	○	○	3	4	4
	Simulation	○	○	○	2	3	3
モニタリング	Hardware monitor	◎	○	×	1	1	1
	Software monitor	◎	○	×	2	2	2

○　は適用可能
×　適用不可能
◎　は同じシステムが利用可能の場合は適用可
１～５は利用の困難な順に
　　コストが高い順に　｝それぞれ５段階にわけた。
　　精度のよい順に

最後にパフォーマンスの測定に関して特に重要な役割りを果すワーク・ロードの問題についてもう少し補足を加え，この項の結びとする。

ワーク・ロードは，実際のシステムを解析するアプローチにあっても又，モデルを用いて解析するアプローチにあっても，共通して重要な位置を占めている。

実際のシステムを解析するアプローチにあっては，どう云う性格の刺激を与えたらシステムのどう云う性格がつかめるかと云う興味から，評価の目的にあったワーク・ロードをいかにして作り出すかと云う形で問題になってくる。

又，モデルによって解析するアプローチに於ては，一般に，ある仮定に基いてモデルの中に組み入れたり，あるいは外部からドライブ・ワーク・ロード（drive work load）として与えてやる必要があるが，いづれにしろワーク・ロードとしてどのような分布や性格を仮定するかが問題となる。

これらの問題を解決する場合にまず重要なことは，実際のワーク・ロードそのものの性格を明らかにすることであって，それにはジョブ・ミックス（Job Mix）の解析，TSS端末ユーザの振舞いの分析，プログラムの振舞い（Program behavior）などの実測にもとづいた研究が必要であり，システムが複雑化するにつれ，益々メジャーメント（measurement）とその分析は重要な要素となりつゝある。

＊ 表2－1は　"計算機システムの評価について"

萩原　宏，京都大学工学部　　情報処理 Vol.13 №11

を参考にしたが個々の項目の数字は目的・対象により異なる場合がある。

2.2 評価手法

この節ではシステム評価のための技法の各々に関して，その詳細を解説し，実施例を中心にその分野での研究の概況を伝える。

A. Add & Cycle Time法

過去においてはCPUの性能が計算機システムの全体を代表する重要なファクタとなっていた。

その当時に於ては，CPUの処理能力が一つの関心事であり，CPUのサイクル・タイムと加算命令の実行時間によってその計算機システムの処理能力を推定しようとする方法が考案された。

即ち，メモリーの読み出し時間，CPUのサイクル・タイム，加算時間が短かい程その計算機システムは優れていると云うことになる。

各種計算機システムについて，そのAdd & Cycle Time 特性を一覧したものとして，KEY DATA 社の一覧リストが名高い。

参考までに表2－2にその一部を掲げておく。この方法は，未だ大部分のプログラムが機械語でコーディングされていた頃に専ら用いられていたが，現在では全くと云って良い程使用されていない。それは，ソフトウェアに対する考慮が全く為されていないこと，又ハードウェアそのものを見積る尺度としても十分なものとは云えないからである。

それには次の様な理由があげられる。

- 計算機の構成を無視している。
- ハードウェアによる命令の先取りの様な特別な機能やワード・サイズ等についても考慮していない。
- 現在の計算機では，加算命令がその計算機システムの特徴を十分に代表しているとは云えない。
- 機械によってはデータのパスも異なり，サイクル・タイムの比較も意味が無い。
- 番地付け方式も無視している。

表2－2　KEY DATA社の一覧リスト

Price Range Monthly in Thousand Dollars	First Delivery Month and Year	Processor Speed Complete add Time in Microseconds	Storage Cycle Time in Microseconds	Accumulators	Internal Storage Capacity in Thousand Words	Word Size	Floating-Point Precision	Overlap	Instruction Set Address Size	Operation Codes	Indirect Addressing	Index Registers	Extensiveness ‡	Time-Sharing ‡
BURROUGHS B500														
1.2-2.0	10/68	170[A]	6	0	9.6-19.2[D]	1a[D]	—	—	18b	28	—	0[H]	XDF	—

A. Assumes two one-character fields. D. Memory organized in six bit characters. L. Indexing done through operand modification. R. 9374-1, 9375-9, 9376-9 also available. T. 9380-2, 9380-3, 9380-1, 9384-3, 9384-4.

BURROUGHS B2500														
4.7-18.2	5/67	64[A]	2[H]	—	5-30	2a[K]	99[F]	—	24	99	∝	3[L]	XD	XP

A. Assumes two five-digit fields. B. Per two bytes. E. Memory is organized in eight-bit bytes or two four-bit digits. F. Decimal digits. L. For each program. P. Up to ten available. T. 9382 and 9390

BURROUGHS B3500														
5.3-27	5/67	32[A]	1[H]	—	5-250	2a[K]	99[F]	—	24	99	∝	3[L]	XD	XP

A, B, E, F, L, T, U, V. See B2500. P. Up to 20 available.

BURROUGHS B5500														
16-100	11/64	2[A]	4	—	4-32	48b	39	—	15	—[J]	∝	0	XBE	XP

A. Instruction look-ahead allows increased internal speed. J. Programs are written in source language. P. Up to four floating channels available. R, U, V. See B283. T. See B300. X. ALGOL in addition to

BURROUGHS B8500, B7500														
25-150	1/68	4[A]	.6	—	16-106[D]	48b	39	—	20	—[J]	∝	0	XBE	XP

A, J, X. See B5500. D. 524 for B7500. Thin-film memory. P. Up to

COLLINS C-8311														
•	•	13	3	0	4	L2b	—	—	3[H]	12	1	—	L	—

H. References eight 12-bit address registers.

COLLINS C-8500														
2.5-3.6	1/67	4.5	2	1	4-65	32b	—	4	18	93	∝	3	BDL	CIMS

T. 8047, 8048, 8049 and 8841A/1 also available. W. ASR 33/35 teletype

COMPUTER PROPERTY MONROBOT XI														
.4-1.0	5/60	6000	12000	1	1-2[D]	32b	—	—	11	27	—	0	BH	—

D. Internal storage is drum. T. CRAM available. U. Card reader/punch is IBM 021 or IBM 026. V. Printer is IBM Model B Typewriter.

Input-Output Number of Channels	Transfer Rate	Auxiliary Storage Fixed Head	Movable Head	Magnetic Tape	Peripheral Devices Card Reader	Card Punch	Printer	Paper-Tape Reader	Paper-Tape Punch	Software Algebraic Compiler	Monitor §	Business Compiler
1	200K	9370[5R]	—	9391[T]	9111[U]	9211[U]	9240[V]	9120	9220	—	G	√

9390 and 9396 also available. U. 9110, 8112 card readers and 9210 card punch also available. V. 9215-2, 9245-3 and 9241 also available.

4[P]	IM	9372	—	9381[T]	9110[U]	9210[U]	9240[V]	9120	9220	√	GRT	√

series also available. U. 9111 and 9112 readers, and 9211 punch also available. V. 9241 and 9242 also available.

6[P]	2M	9372	—	9381[T]	9110[U]	9210[U]	9240[V]	9120	9220	√	GRT	√

1[P]	•	B430[R]	B475	B421[T]	B122[U]	B303[U]	B320[V]	B141	B341	√[X]	GRT	√

FORTRAN. Note. All B5000 systems have been field-converted to B5500 systems.

4[P]	•	9372	—	9381[T]	9110[U]	9210[U]	9240[V]	9120	9220	√[X]	GRT	√

eight floating channels available. T. U. V. See B2500.

—	—	—	—	B845	—	—	—	—	—	—	—	—

32	5.1M	8873	8871	8046[T]	8861	8862	8852	—[W]	—[W]	√	—	—

available.

4	•	—	—	—[T]	—[U]	—[U]	—[V]	—[W]	—[W]	√	—	√[Z]

W. Paper tape equipment is made by Royston. Z. Quikomp.

†X – all except: B – byte manipulation, D – double precision, E – translate edit capability, F – floating point instructions, M – hardware multiply-divide, L – logical operations
‡X – all except: A – base address relocation, C – clock, I – program interrupt, M – memory protection, P – dynamic page relocation, S – supervisor mode.
§G – batch, R – real-time, T – time-sharing.
— None. ▽ See Section II-B • Information unavailable.

CENTRAL PROCESSORS CHARACTERISTICS

B. インストラクション・ミックス法

Add & Cycle Time による方法が，加算命令の実行時間だけでそのハード特性を代表しようとしたのに対し，この方法は，ある分野に於ける命令の実行頻度の分布から各命令に与える重みを定め，一命令あたりの平均実行時間を求めて，それを評価の基準とする方法である。

例えば，ある命令セットの中から n 個の命令を取り出しこの命令群に W_i の比重を与えるなら，$t_1 \sim t_n$ を各命令の実行時間として，インストラクション・ミックス M は次式で表わされる。

$$M = \Sigma W_i t_i$$

この方法は，何種類かの命令を考慮に入れている点で前者よりは優れた方法であると云える。

しかし前者と同様の欠点もやはり指摘される。即ち，ワード・サイズ，データのパス，番地付け方式，ソフトウェア，入出力構成等に関して何等考慮が払われていないことである。

又各命令に与える重みについても，それを決める基準は，はっきり定まっているわけではなく，いくつかのプログラムのトレースを行って得られた結果からこれを決定している。

しかしそれが妥当であるという保証は無い。

例えば，ある例ではFORTRANで書かれた逆行列の計算プログラムをトレースして表2－3で示される様な命令比率が得られたと云う。

これはGibsonの与えた結果，表2－4とかなりの違いを示していることが分る。この結果得られるインストラクション・ミックス値もGibson Mixの約1.1倍になることが分る。

この他にもこの方法による欠点として次の様な点が挙げられている。

- ○命令の集合は計算機によって異るが，それにもかかわらず同一の方法で重みずけを行うことになる。
- ○同一の仕事を異った命令を使って処理した時の効果を示すことも困難で

ある。

しかし以上の様な事情を十分承知の上で，なおハードウェア性能の指標の一つとして現在でも広く用いられている。

以下，対象とする分野によって各種のミックスが考えられるが，現在比較的よく使われているものについてのみ簡単に説明を加えておく。

(1) Gibson Mix

ミックスの中でも最もなじみの深いもので，科学技術用計算機の性能の指標として現在最もよく使用されている。

表２－３　実験的に求めた荷重の一例

	instructions	weight
1	add／sub／load／store	6.0％
2	multiply	11.4
3	divide	0.0
4	branch	10.7
5	compare	9.6
6	transfer	0.0
7	shift	0.0
8	And／Or	0.0
9	index	34.5
10	short floating add／sub	29.0
11	long floating add／sub	0.0
12	short floating multiply	8.6
13	long floating multiply	0.0
14	short floating divide	0.1
15	long floating divide	0.0

その名の由来はJ. C. Gibson氏にあると云われている。

Gibson Mixは，インストラクション・セットの中から，13群のカテゴリーに属する命令を取り出し，各カテゴリーにある荷重をかけて平均をとった平均命令実行時間である。

荷重の決定には，７種類の科学技術計算のプログラムについてトレースをとり，各命令の出現回数の統計をとったと云う。

この結果を表２－４に示す。

現在我が国で広く利用されているGibson Mixは，これと若干異るもので表２－５で示されるものが使用されている。

表２－４　Gibson Mixの荷重

	Instructions	Weight
1	load and store	31.2 %
2	fixed point add and subtract	6.1
3	compare	3.8
4	branch	16.6
5	floating add and subtract	6.9
6	floating multiply	3.8
7	floating divide	1.5
8	fixed point multiply	0.6
9	fixed point divide	0.2
10	shifting	4.4
11	logical, And, Or, etc.	1.6
12	instructions not using registers	5.3
13	indexing	18.0

表２－５　一般に使用されている荷重

	Instructions	Weight
1	add／sub／load／store	33.6 %
2	multiply	0.6
3	divide	0.2
4	branch	6.5
5	compare	4.0
6	transfer	17.5
7	shift	4.6
8	And／Or	1.7
9	index	19.0
10	short floating add／sub	5.8
11	long floating add／sub	1.5
12	short floating multiply	3.2
13	long floating multiply	0.8
14	short floating divide	1.3
15	long floating divide	0.3

(2) コマーシャル・ミックス

一方，事務用計算機の性能を比較するのに用いられるミックスとしてコマーシャル・ミックス（Commercial Mix）がある。前者が演算命令に重点を置いているのに対しこれは編集，表引き，データ転送等に重きを置いたものである。

一般に表２－６の様な荷重が使用されている。

表２－６　コマーシャル・ミックスの荷重

	Instructions	Weight
1	arithmetic operation	9 %
2	compare	24
3	move	25
4	jump	31
5	edit	4
6	I/O initiation	7

(3) リアルタイム・ミックス（Real Time Mix）

オンライン・リアルタイム用のコンピュータの性能評価に最近使用されているのにリアルタイム・ミックスと呼ばれるものがある。

一般に表２－７の様な荷重が使用されている。

表２－７　リアルタイム・ミックスの荷重

	Instructions	Weight
1	load	22 %
2	store	22
3	add/sub	11
4	conditional jump	9
5	unconditional jump	8
6	compare	6
7	logical	6
8	shift	6
9	index increment and test	5
10	loader modify index	5

(4) その他のミックス

その他，コマンド・アンド・コントロール・システムの性能評価に用いられるミックスとか，FORTRANやCOBOLのコンパイル時の平均命令実行時間を求めたミックス等，各種のミックスが考案されている。

又コンパイル，リンク，実行までの過程を考慮して平均命令実行時間を求めようとする方法（複合評価法）等も提案されている。

参考までにその一例を掲げておく。

複合評価法の例

E_{SH} ： 科学技術計算処理の時の実効的1ステップ実行時間

T_C ： コンパイル時の命令ミックス

T_L ： 結合編集時の命令ミックス

T_E ： 実行時の命令ミックス（≒GIBSONミックス）

W_C ： コンパイルの比率
W_L ： 結合編集の比率
W_E ： 実行の比率
（$W_C + W_L + W_E = 1$）

S_C ： コンパイルに対するサンプル・プログラムのステップ数による補正係数

$$\frac{\text{目的機種のステップ数}}{\text{標準機種のステップ数}}$$

S_L ： 結合編集に対するサンプル・プログラムのステップ数による補正係数

S_E ： 実行に対するサンプル・プログラムのステップ数による補正係数

$$E_{SH} = W_C \cdot S_C \cdot T_C + W_L \cdot S_L \cdot T_L + W_E \cdot S_E \cdot T_E$$

又，K. E. Knight は計算機の能力をComputing Power Pとして表現することを提案した。

これは単にインストラクション・ミックスだけで性能を見積るのでなく，さらに重要な要素として，入出力特性，主記憶容量等を考慮して計算機の能力を表現しようと試るものである。

参考までにPの算出法を掲げておく。

$$P = \frac{M \times 10^{12}}{t_C + t_{i10}}$$

t_C は次式で与えられる。

$$t_C = C_1 A_{F_i} + C_2 A_{FL} + C_3 M + C_4 D + C_5 L$$

A_{F_i} ：固定小数点加算時間

A_{FL} ：浮動小数点加算時間

M ：乗算時間

D ：除算時間

L ：論理演算時間

$C_1 \sim C_5$ はそれぞれ荷重で，科学計算，事務計算で異なり，表２－８の様にきめられている。

表２－８ Knight の荷重

Instructions	Weight	
	sccentific	commercial
1 fixed add		
a. for computers without index registors or indirect addressing	10 %	25 %
b. for computer with index registers or indirect addressing	25	45
2 floating add	10	0
3 multiply	6	1
4 divide	2	0
5 logical operation		
a. for computers without index registers or indirect adressing	72	74
b. for computers with index registers or indirect adressing	57	54

C. カーネル・プログラム（Kernel Program）

カーネル法とは，あるプログラム全体，或いはその核となる主要な部分だけを実際に対象とする計算機の機械語でコーディングし，その実行時間を見積る方法である。実行時間の計算には，計算機メーカの発表しているマニュアル・データをもとにして各命令ステップ毎に要する時間を積算してゆく方法をとる。

カーネル・プログラムとして使用するサンプル・プログラムとしては例えば科学計算では逆行列，ベクトル積，多元連立方程式，各種関数副プログラム等が考えられる。

同様に事務計算の場合は給与計算等が考えられる。この方法によれば，番地付け方式，特別なインデックス・レジスタ等その機械に固有な特性も考慮できるし又広範囲のインストラクション・セットを利用することも可能になり，前2者と比較してさらに優れた方法になる。

しかしカーネル法に於ても，入出力オペレーションに関する十分な考慮が為されていないこと又オペレーティング・システム等，ソフトウェアの効果が十分に評価できないことが欠点としてあげられる。

その他，いくつものカーネル・プログラムを作成する時を考えると，実行時間の見積りにはかなりの労力を要し大変であること又各機種について同程度のプログラミング技法を使用しなければ正しい評価ができない等の問題があげられている。

カーネル・プログラムの最も良く知られたシリーズとして，Auerbach EDP Standardsがある。

（注．Auerbach はこれらのカーネル・プログラムをベンチマークと呼んでいる）

Auerbach では次に示す5つのカーネル・プログラムを標準的問題として，その実行に要する時間を測定し，いくつかの計算機システムについてパフォーマンス・チャートを得ている。

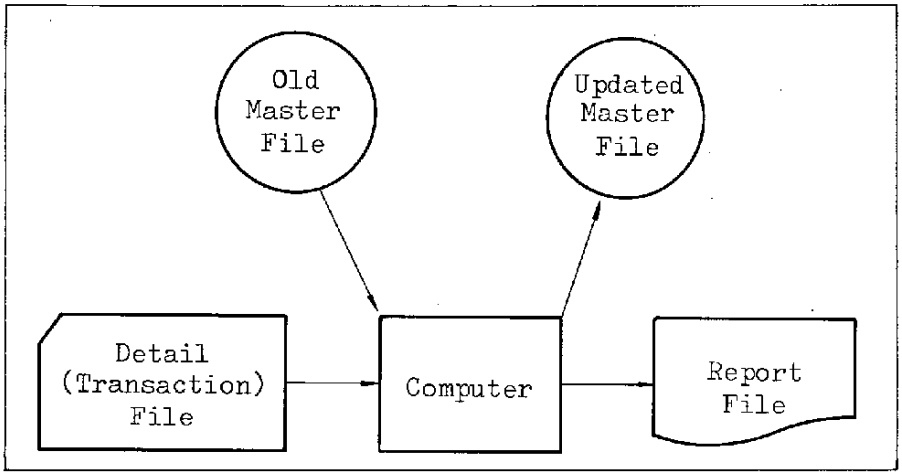
表2－9にその一部を掲げておく。

〔Auerbach Standards が扱う標準プログラム〕

① Generalized File Processing Problem A

（一般のファイル処理に関する問題A）

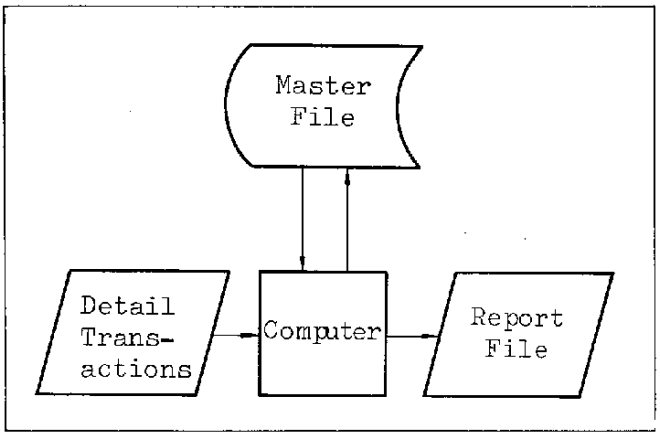
トランザクション・レコードにより旧マスタ・ファイルを更新して新マスタ・ファイルを作成するごく一般的なファイル処理の過程である。



② Random Access File Processing

（ファイルの乱処理）

リアルタイム・システムのアプリケーションを代表するもので，オンライン・マスタ・ファイルに対するインクウィアリに答えたり，色々なタイプのトランザクションによって修正を行ったりする過程を表現したプログラムである。



③ Sorting（ソーティング）

レコードのソーティングを行う過程を表現したプログラムである。

④ Matrix Inversion（逆行列の計算）

10×10及び40×40のマトリックスの逆行列を求めるプログラムである。

（メーカ供給のルーティンがあれば，表のAvailableの項にその場合の時間も求めている）

⑤ Generalized Mathematical Problem A

（一般的な数値問題A）

データを読み，次のタイプの多項式を計算して結果をプリントするプログラムである。

$$Y = A + BX + CX^2 + DX^3 + EX^4 + FX^5$$

表2－9　Auerbach Standardsの一部

SYSTEM IDENTITY			GENERALIZED FILE PROCESSING PROBLEM A (minutes per 10K records)			RANDOM ACCESS BENCHMARK PROBLEM		SORTING PROBLEM (10K 80-char records)	
			Activity Factor			Timing Summary		Timing in Minutes	
Manufacturer, Model Number	Standard Configuration Number	Monthly Rental	0.0	0.1	1.0	Time per Transaction (msec)	Time per 10K Records (min)	Standard Estimate	Available Routines
Burroughs B 500	I	3,060	—	—	67	—	—	—	—
	II	3,955	2.2	2.9	19	—	—	22	—
	III	5,950	1.4	2.5	19	—	—	9.5	11
Burroughs B 2500	II	4,950	0.95	1.9	18	—	—	7.5	—
	III	6,455	0.75	1.8	18	—	—	5.0	—
	IVR	9,210	—	—	—	125.	21.	—	—
Burroughs B 3500	IVR	11,064	—	—	—	125.	21.	—	—
	VIIA	15,999	0.37	2.0	18	—	—	2.5	—
	VIIIR	19,470	—	—	—	301.	50.	—	—
Burroughs B 5500	III	26,003	1.2	2.0	19	—	—	—	—
	V	26,353	1.2	2.0	19	—	—	—	—
	VIIA	32,835	0.55	1.7	17	—	—	2.9	2.8
	VIIB	30,860	0.55	0.69	1.8	—	—	2.9	2.8
CDC 3100	IIIR	8,715	—	—	—	—	—	—	—
	IVR	12,805	—	—	—	—	—	—	—
	VI	13,395	0.94	2.0	20	—	—	6.1	—
	VIIA	16,085	0.36	2.0	20	—	—	2.7	—
CDC 3300	IVR	14,530	—	—	—	—	—	—	—
	VI	14,885	0.94	2.0	20	—	—	6.1	—
	VIIA	19,535	0.36	2.0	20	—	—	2.7	—
CDC 3400	VI	14,930	0.56	1.96	16	—	—	3.7	—
	VIIA	18,460	0.56	1.62	16	—	—	3.7	—
	VIIB	19,195	0.56	0.77	2.6	—	—	3.7	—
	VIIIB	32,030	0.29	0.33	1.0	—	—	1.8	—
CDC 3600	VIIA	34,575	0.19	0.28	1.2	—	—	1.4	—
	VIIB	34,530	0.19	0.28	1.2	—	—	2.0	—
	VIIIB	48,115	0.19	0.19	1.0	—	—	1.4	—
CDC 6400	VIIA	38,215	0.38*	0.38*	2.0*	—	—	2.5	—
	VIIIA	49,415	0.19*	0.19*	1.0*	—	—	1.3	—
CDC 6600	VIIA	57,600	0.38*	0.38*	2.0*	—	—	2.5	—
	VIIIA	68,880	0.19*	0.19*	1.0*	—	—	1.3	—
GE 415	I	5,405	—	—	75	—	—	—	—
	II	7,420	2.4	2.4	15	—	—	24	—
	III	8,790	1.8	1.8	15	—	—	13	—
	IV	14,835	0.47	1.5	15	—	—	3	—
	VIIA	16,205	0.47	1.5	15	—	—	3	—
GE 425	I	6,435	—	—	61	—	—	—	—
	II	8,450	2.4	2.4	15	—	—	25	—
	III	9,820	1.8	1.5	15	—	—	13	—
	IV	15,865	0.47	1.4	15	—	—	3.1	—
	VIIA	17,565	0.47	1.4	15	—	—	3.1	—
GE 435	III	11,910	1.8	1.8	15	—	—	13	—
	IV	17,955	0.47	1.4	15	—	—	3.1	—
	VIIA	18,665	0.47	1.4	15	—	—	3.1	—
GE 635	VIIA	42,660	0.47*	0.70*	2.8*	—	—	3.1	—
	VIIIA	58,265	0.26*	0.26*	1.4*	—	—	1.7	—

*Indicated time is for the tape-to-tape main processing run only; it is assumed that the required on-line card-to-tape and tape-to-printer transcriptions will be performed with these or other programs.

表２－９の続き

SYSTEM IDENTITY		MATRIX INVERSION (time in min)				GENERALIZED MATHEMATICAL PROBLEM A (time in msec)		
		Standard Estimate		Available Routines				
Manufacturer, Model Number	Standard Configuration Number	Size of Array				Computation Factor for 10 Percent Output		
		10×10	40×40	10×10	40×40	1	10	100
Burroughs B 500	I	—	—	—	—	—	—	—
	II	—	—	—	—	—	—	—
	III	—	—	—	—	—	—	—
Burroughs B 2500	II	—	—	—	—	—	—	—
	III	0.026	1.5	—	—	78	700	6,900
	IVR	—	—	—	—	—	—	—
Burroughs B 3500	IVR	—	—	—	—	—	—	—
	VIIA	0.013	0.75	—	—	78	350	3,500
	VIIIR	—	—	—	—	—	—	—
Burroughs B 5500	III	0.0025	0.14	0.006	0.25	74	74	330
	V	0.0025	0.14	0.006	0.25	74	74	330
	VIIA	0.0025	0.14	0.006	0.25	74	74	330
	VIIB	0.0025	0.14	0.006	0.25	9.5*	39*	330
CDC 3100	IIIR	—	—	—	—	—	—	—
	IVR	—	—	—	—	—	—	—
	VI	0.0013	0.08	—	—	50	50	330
	VIIA	0.0013	0.08	—	—	50	50	330
CDC 3300	IVR	—	—	—	—	—	—	—
	VI	0.0008	0.046	—	—	50	50	265
	VIIA	0.0008	0.046	—	—	50	50	265
CDC 3400	VI	0.0004	0.026	—	—	65	65	145
	VIIA	0.0004	0.026	—	—	65	65	145
	VIIB	0.0004	0.026	—	—	12	23	145
	VIIIB	0.0004	0.026	—	—	9.9	23	145
CDC 3600	VIIA	0.0003	0.017	—	—	6.0	6.5	61
	VIIB	0.0003	0.017	—	—	6.0	6.5	61
	VIIIB	0.0003	0.017	—	—	6.0	6.5	61
CDC 6400	VIIA	0.00022	0.011	—	—	13*	13*	13*
	VIIIA	0.00022	0.011	—	—	6.2*	6.2*	6.2*
CDC 6600	VIIA	0.00003	0.0014	—	—	13*	13*	13*
	VIIIA	0.00003	0.0014	—	—	6.2*	6.2*	6.2*
GE 415	I	—	—	—	—	—	—	—
	II	—	—	—	—	—	—	—
	III	—	—	—	—	—	—	—
	IV	—	—	—	—	—	—	—
	VIIA	0.0029	0.17	—	—	120	280	1,800
GE 425	I	—	—	—	—	—	—	—
	II	—	—	—	—	—	—	—
	III	—	—	—	—	—	—	—
	IV	—	—	—	—	—	—	—
	VIIA	0.0021	0.12	—	—	100	240	1,400
GE 435	III	—	—	—	—	—	—	—
	IV	—	—	—	—	—	—	—
	VIIA	0.0016	0.09	—	—	74	190	1,300
GE 635	VIIA	0.0004	0.021	—	—	13*	14*	113*
	VIIIA	0.0004	0.021	—	—	8*	14*	113*

*Indicated time is for the tape-to-tape main processing run only; it is assumed that the required on-line card-to-tape and tape-to printer transcriptions will be performed with these or other programs.

D. ベンチマーク・プログラム（Benchmark Program）

対象とする計算機システムの性格を知るための最も直感的な方法として，実際にテスト・プログラムを作成し，その機械で実行させて計算時間を測定するという方法が考えられる。

この様に，システムの性能をテストする目的で作成され，実行されるテスト・プログラムのことをベンチマーク・プログラムと呼んでいる。

例えば，あるコンパイラーのオブジェクト効率を調べようとするならば，同一の問題に対し一方は高級言語で，もう一方はアセンブラー言語で記述した2つのベンチマーク・プログラムを用意し，両者の実行時間の相違を検討すればよいであろう。

この様に，ベンチマーク法では評価しようとする対象によって適切なテストプログラムを用意すれば単にハードウェア性能の比較だけでなく，例えば入出力装置のパフォーマンス評価，各種ソフトウェア・パッケージの性能評価，OS の性能評価等，広範囲な応用が可能になってくる。

又 Job Mix を代表する様な一連のベンチマークを用意すればOSマルチ・プログラミング下に於ける処理効率等も測定することができる。

ベンチマーク法の実際の利用面としては，例えば次の様な場合が考えられるだろう。

① 機種選択のためにいくつかの計算機システムの性能を比較する必要がある場合。

② センタ運営等に於て，機器構成の最適化を計ってゆくような場合（機器構成の変更による効率の変化を計測する必要がある）。

③ システム・プログラムの改善を行ってゆく時（その前後で改善が実現されたかどうかを調べる必要がある）。

④ 各種ソフトウェア・パッケージの比較を行う場合。

小さなテストプログラムも含めて考えるならその応用範囲は非常に広く，これまでにも多くの実施例が報告されている。

例えば，ハードウェア性能の測定に関するものでは Raichelson，Ar-

buckle等の報告がある。

又システム・パフォーマンスの測定に関してはBackley，Joslin等の報告がある。

Dopping，Williams 等は4～5ケの標準的なベンチマーク・プログラムを作成し，いくつかの計算機システムの性能評価を行った例を報告している。

Rubey 等はPL／IとCOBOL，FORTRAN，JOVIAL 等の言語比較を行うために表2－10の様な7つの問題を用意し，同じプログラマーに同じ問題を2つの言語でコーディングさせ測定を行った例を報告している。

表2－10　Rubeyの用いたベンチマーク・プログラム

1. GROSS PAYROLL（通常の給料計算）
2. ALOREP 2（航空積荷問題）
3. MMI（Man／Machine Interaction）
4. TSME（Throughput Simulation in a Multiprogramming Environment）
5. VIG（Vehicle Impact and Guidance）
6. SPP—A（Simulation Post Processor—A）
7. SPP—B（Simulation Post Processor—B）

亀田等はCONTESTと呼ぶシステム（テストプログラム・ジェネレータ，後述するシンセティック・プログラムに近い）を開発し，各種入出力パターンを持ったFORTRANプログラムを生成して入出力オペレーションに於ける処理効率の測定を行った。又STOP END ジョブによるスループットの測定も併せて行った。

この他TSSの評価に関するもの等，数多くの報告がある。

ここでベンチマーク法の利用に関して特に留意しなくてはならない点をあげると次の様になる。

① Job Mix の表現が困難

多重プログラミング・システムに於ける JOB 処理効率の評価を行う様な場合には現実の Job Mix をできるだけ反映する様なベンチマーク・プログラムを多数用意しなければならないが，その様な Job Stream を作ること自体が仲々困難である。

② 数多くのベンチマーク・プログラムを作成する必要がある

少数のベンチマーク・プログラムによる結果から評価するのは危険である。

例えば Joslin らの行ったテストによれば 4 つのシステム A，B，C，D に 4 つのベンチマーク・プログラム W，X，Y，Z を走らせた結果，実行時間について表 2－11 の様な結果を得たと云う。

表 2－11　Joslin の行ったテスト結果

システム / 問題	A	B	C	D
W	1.00	1.10	2.10	2.57
X	1.36	1.00	2.09	2.00
Y	1.00	1.32	2.72	1.35
Z	1.12	1.00	2.12	4.05

この例からも明らかなようにシステム（A，B），（C，D）の 2 グループ間の順位は変わらないがグループ内では問題毎に順位が変わっていることが分る。

この様に少数のベンチマーク・テストの結果から結論を下すのは危険であり，数多くの試みから十分に検討する必要がある。

③ システム・パフォーマンスの評価は困難

システム・パフォーマンスを評価する場合もいくつかのプログラム・クラスに対して十分慎重な検討を行った上，各クラスに関する適当な重み付けを決定する必要がある。

これには多くのプログラム統計からその重みを決定することが必要になろう。

例えば Dopping，Williams 等は次の様な重みを採用している。

④ ソフトウェアの比較にはその設計目標の相違にも十分注意を払わなくてはいけない。

例えばコンパイラ等の比較を行う時，そのコンパイラがコンパイル効率に重点を置いて作成されたか等について考慮しなくては比較は意味が無いことになる。

表2－12　Doppingの採用した重み

	重み
1. Card to MT(Magnetic Tape)変換	20%
2. Sorting	40
3. MTからのprintout	40
4. 入出力	
5. CPUと主記憶装置の性能テスト	

表2－13　Williamsの採用した重み

1. Marketing information retrieval and mailing list preparation	35%
2. Billing and sales analysis	30
3. Scientific information retrieval	30
4. Statistics	5

以上述べた様な点を考慮すると，評価対象を限っても標準的なベンチマーク・プログラムと云うのは仲々考えられないかもしれない。

この点に関してH·Lucas等は次の様に述べている。

ベンチマークの中には個々の利用者のためにそれぞれプログラミングしなければならないようなものが多数ある。そこで各メーカーの機械について総合テストのモジュールの標準的な集合を用意しておいて，その中から適当に選択して使えるようになっていれば，パフォーマンスの評価や比較がひじょうに容易になる。

そのような総合テストのプログラムをつくるときは，広範囲にわたる特徴をもつようなものでなければならない。各総合テストのプログラムはモジュールの集まりで構成する。それらのモジュールはコンパイルと実行，ハードウェア機能，入出力の機能，およびバッチやリアルタイム・オペレーティング・システムをテストするように設計する。小型の処理装置を置いてそれから，オンライン・システムにメッセージを送るような多重計算機システムの

場合には，総合テストのプログラムと各機械との間のインターフェースで問題が起る。各モジュールの正しい内容は標準をつくるグループが規定しなければならない。そのときに含めなければいけない項目の一部が表2－14に示してある。

表2－14　総合テストプログラムの機能的な分類

コンパイルと実行	
処理時間： 　計算制約のルーチン 　入出力制約のルーチン 　入出力と計算とを含むルーチン 　ライブラリとサブルーチンのテスト 実行の融通性： 　データの転送，ビットやバイトの操作 　入出力操作の範囲 　更新 　並行処理 　セグメンテーション，オーバーレイ	エラー： 　コンパイル時のエラー 　実行時のエラー
オペレーティング・システム	
ジョブ管理： 　ソースデック，コンパイル，実行 　コンパイル 　ロードと実行 データ管理： 　テープファイルの処理 　ディスクアクセスの方法 遠隔通信： 　伝送ルーチン 　バッファリングとメッセージの処理 エラー： 　ハードウェアとソフトウェアのエラーの検出 　実行エラーと入出力エラー	制御システム： 　入出力の制御と割付け 　記憶域の割付け 　周辺装置の操作 　スプーリング 　ライブラリ 　アカウンティング・システム 　ジョブからジョブへの移行 　優先順位づけ その他の特徴： 　並列ジョブ 　多重プログラミング，多重処理 　端末システム 　再スタートとエラーの回復

又，亀田等は次の様な標準プログラムを用意し実験を行ったことを報告ししている。

表２－15　FORTRAN処理システムの比較テスト

1. STOP END job（５個）
2. CONTEST プログラム（入出力テスト）
3. エラー・メッセージ・テスト・プログラム
4. 文法テスト・プログラム
5. 各 FORTRAN 文の実行測定テスト
6. 最適化テスト（ FOPTEST ）
7. 計算処理速度テスト

最短距離，行列の積，数値積分，多重回帰分析，行列の積については，５通りの解法を，次元を４通りにかえてテストした[36)]

表２－16　機種選択のためのテスト

1.	STOP END job	５個
2.	数ステートメント・ジョブ	５個
3.	数十ステートメント・ジョブ	20個
	（そのうち10個は文法的誤りをもつ）	
4.	倍長精度検定プログラム	
5.	エラー・メッセージ・テスト・プログラム	
6.	CPU性能テスト 行列の固有値，行列の積，積分，輸送問題	

E. シンセティック・プログラム（Synthetic Program）

シンセティック・プログラムもベンチマーク・プログ　ムと同様に実際にコーディングを行い計算機で実行する一種のテスト・プログラムである。

前者との違いは，シンセティック・プログラムではパラメータの指定によってプログラムの性格を自由に変えることが出来ること，又ベンチマーク・プログラムがその内容，結果に関してプログラムとして意味があるものであったのに対しシンセティック・プログラムではプログラムとして特に意味を持つ必要が無いという点である。

パラメータの指定だけで要求にあった性格のプログラムが編成できると云うことは色々な点で便利である。

例えば入出力動作に関する効率測定を行う様な場合には，ベンチマーク法によれば各種の入出力パターンを持ったプログラムをいくつか用意しなくてはならない。これがシンセティック・プログラムとして準備されれば，同一のプログラムのパラメータを変えるだけの作業で済むことになる。

又，多重プログラミング下に於けるOSのパフォーマンスを測定する様な場合も各種の性格を備えたJobをいくつか用意し，適当な混合を行ってテスト・ジョッブ・ストリームを編成する必要がある。

この様な場合もシンセティック・プログラムは効果的に利用されるであろう。

この様にシンセティック・プログラムは柔軟性と云う点でベンチマーク・プログラムより優れた性格を備えている。

これはテスト・プログラム作成に要する労力を大いに軽減してくれることになる。

その応用範囲，問題点等については大体ベンチマークと同様なことが云えるが特にJob Mixの表現と云う点では大きな効果を発揮するだろう。

又，Buchholz等はシンセティック・プログラムに要求される事項として次の様な点を指摘している。

「よいシンセティック・プログラムを作成するには，さまざまな利用者に

よって要求される各々特徴ずけられたプログラムの編成を可能にしなくてはならない。これにはその表現する内容が広範囲にわたることが必要である。又，異機種間での比較を可能にするためにはマシン・インディペンデントなプロセデュアであることが望ましい。

その他，計算機システムの規模が変わっても問題が無いようにするためにはその処理内容は余り複雑になることは好ましくない。又処理時間に対する考慮等も要るだろう」

シンセティック・プログラムについてもいくつかの実施例が報告されているが，例えばBuchholz の考えたシンセティック・プログラムは次の様に構成されている。

（Buchholzのシンセティック・プログラム）

プログラムの処理内容は一般のデータ・プロセシングの中核となるファイル・メインテナンス・プロセデュアをモデル化したものである。

外部記憶装置との入出力操作に重点を置く部分と，計算に重点を置くコンピュート・カーネルから構成され，パラメータの指定でその比率が自由に変えられる様になっている。

以下簡単にプログラム動作について述べる。

「インプットとしてディテール・レコードが与えられる。マスター・ファイルを探してマッチするレコードが見つかったら一連のコンピュート・カーネルを実行する。実行が終了したらマスタ・レコードを更新し，マスター・レコードとディテール・レコードを書き出す」

以上の過程が繰り返されると云う単純なプロセスであるが，ここでマスター・レコードの数，ディテール・レコードの数等はパラメータで指定できる様になっている。

又，コンピュート・カーネルとI／O の比率，所要記憶容量等も指定できる様になっている。

参考までにそのプログラム・リストと結果の一部を図２－３に掲げておく。

```
YSTKP: PROCEDURE OPTIONS (MAIN);                                        1
   /* THE FOLLOWING DECLARE STATEMENT IS IMPLEMENTATION-DEPENDENT */    2
   DECLARE PARAMS FILE INPUT,                                           3
        MASTER FILE RECORD INPUT,   DETIN  FILE RECORD INPUT,           4
        NEWMAS FILE RECORD OUTPUT,  DETOUT FILE RECORD OUTPUT,          5
        MASGEN FILE RECORD OUTPUT,  DETGEN FILE RECORD OUTPUT;          6
   /* THE FOLLOWING STATEMENT DEFINES THE TYPE OF KERNEL ARITHMETIC */  7
   DECLARE (START,SUM,TABLE(1000)) BINARY FIXED(31) STATIC;             8
   DECLARE (I,J,K,N,U,CHECK,COUNT,LSUM,NMAS,NMAS1,NDET,NDET1,NREP)      9
        BINARY FIXED(31),CARD CHARACTER(80),TEMPC CHARACTER(6)) STATIC, 10
        (INTKEY PICTURE'(6)9',KRETURN LABEL,                            11
        (START_TIME,END_TIME) CHARACTER(9)) STATIC,                     12
        1  MASTER_REC ALIGNED STATIC,                                   13
          2  MASTER_KEY CHARACTER(12),                                  14
          2  MASTER_SUM   BINARY FIXED(31),                             15
          2  MASTER_CHECK BINARY FIXED(31),                             16
          2  MASTER_DATA (15) CHARACTER(12),                            17
        1  DETAIL_REC ALIGNED STATIC,                                   18
          2  DETAIL_KEY CHARACTER(12),                                  19
          2  DETAIL_SUM   BINARY FIXED(31),                             20
          2  DETAIL_CHECK BINARY FIXED(31),                             21
          2  DETAIL_DATA (15) CHARACTER(12);                            22
                                                                        23
   OPEN FILE (PARAMS);  ON ENDFILE(PARAMS) GO TO EOF;                   24
   N = 10;  START = 100;  NMAS = 0;  NDET = 0;                          25
   DO J = 1 TO N**3;  TABLE(J) = START + J - 1;  END;                   26
                                                                        27
START_PASS:                                                             28
   GET FILE(PARAMS) EDIT (CARD) (A(80));                                29
   PUT EDIT (CARD) (SKIP(2),A(80));                                     30
   IF SUBSTR(CARD,1,6) ¬= ' PASS ' THEN GO TO START_PASS;               31
   GET STRING (CARD) EDIT (NMAS1,NDET1,NREP) (X(6),3(X(6),F(6)));       32
   IF NMAS1 < 0 THEN GO TO START_PASS;  COUNT = 0;  CHECK = 0;          33
                                                                        34
/* MASTER GENERATION */                                                 35
   IF NMAS1 = NMAS | NMAS1 ¬> 0 THEN GO TO DETAIL_GENERATION;           36
   NMAS = NMAS1;                                                        37
   OPEN FILE(MASGEN);  DO J = 1 TO NMAS;                                38
        MASTER_SUM = 0; INTKEY = J; MASTER_KEY = '000000' || INTKEY;    39
        CHECK = CHECK + J;  MASTER_CHECK = CHECK;                       40
        TEMPC = INTKEY;  MASTER_DATA = 'MASTER' || TEMPC;               41
        WRITE FILE(MASGEN) FROM (MASTER_REC);                           42
   END;  CLOSE FILE(MASGEN); CHECK = 0;                                 43
                                                                        44
DETAIL_GENERATION:                                                      45
   IF NDET1 = NDET | NDET1 ¬> 0 THEN GO TO TAPE_PASS;                   46
   NDET = NDET1; RATIO = NMAS / NDET;                                   47
   OPEN FILE(DETGEN);  DO J = RATIO TO NMAS BY RATIO;                   48
        DETAIL_SUM = 0; INTKEY = J; DETAIL_KEY = '000000' || INTKEY;    49
        CHECK = CHECK + J;  DETAIL_CHECK = CHECK;                       50
        TEMPC = INTKEY;  DETAIL_DATA = 'DETAIL' || TEMPC;               51
        WRITE FILE(DETGEN) FROM (DETAIL_REC);                           52
   END;  CLOSE FILE(DETGEN); CHECK = 0;                                 53
                                                                        54
TAPE_PASS:                                                              55
   IF NREP = 0 THEN GO TO START_PASS;                                   56
   IF NMAS1 ¬> 0 | NDET1 ¬> 0 THEN GO TO COMPUTE_PASS;                  57
   KRETURN = WRITE_DETAIL;                                              58
   OPEN FILE(MASTER), FILE(NEWMAS), FILE(DETIN), FILE(DETOUT);          59
   ON ENDFILE(MASTER) GO TO END_TPASS;                                  60
   ON ENDFILE(DETIN) GO TO RUNOUT;                                      61
   READ FILE(MASTER) INTO (MASTER_REC);                                 62
   READ FILE(DETIN) INTO (DETAIL_REC);                                  63
   START_TIME = TIME;                                                   64
                                                                        65
KEY_TEST:  IF MASTER_KEY < DETAIL_KEY THEN GO TO WRITE_MASTER;          66
   IF MASTER_KEY > DETAIL_KEY THEN DO;  PUT EDIT                        67
        ('SEQUENCE ERROR HALTED RUN')(SKIP,A); GO TO CLOSE_FILES; END;  68
                                                                        69
KERNEL:                                                                 70
/* KERNEL SUMS  N  INTEGERS FROM A TABLE OF  N**3  CONSECUTIVE INTEGERS 71
   BEGINNING WITH 'START'.  K-TH INTEGER SUMMED IS 'START - 1 + K**3'.  72
   SUM IS CHECKED ALGEBRAICALLY.  KERNEL IS REPEATED  NREP  TIMES. */   73
   DO I = 1 TO NREP; SUM = 0; U = 0; J = 0;                             74
        DO K = 1 TO N;  J = J + (6*U + 1);                              75
        SUM = SUM + TABLE(J);  U = U + K;  END;                         76
     LSUM = (N * (N + 1)) / 2;                                          77
     IF START ¬= (SUM - LSUM * LSUM) / N + 1 THEN DO;  PUT EDIT         78
        ('COMPUTE ERROR HALTED PASS')(SKIP,A);  GO TO START_PASS;  END; 79
   END;  GO TO KRETURN; /* KRETURN IS EITHER WRITE_DETAIL OR CRETURN */  80
                                                                        81
WRITE_DETAIL:  MASTER_SUM = SUM; DETAIL_SUM = SUM;                      82
        CHECK = DETAIL_CHECK; COUNT = COUNT + 1;                        83
   WRITE FILE(DETOUT) FROM (DETAIL_REC);                                84
   READ FILE(DETIN) INTO (DETAIL_REC);                                  85
                                                                        86
WRITE_MASTER:  WRITE FILE(NEWMAS) FROM (MASTER_REC);                    87
   READ FILE(MASTER) INTO (MASTER_REC);  GO TO KEY_TEST;                88
                                                                        89
RUNOUT:  DETAIL_KEY = HIGH(12);  GO TO WRITE_MASTER;                    90
                                                                        91
END_TPASS:  END_TIME = TIME;                                            92
   IF CHECK = (COUNT * (COUNT + 1) * RATIO) / 2 THEN GO TO CLOSE_FILES; 93
   PUT EDIT (' CHECKSUM ERROR HALTED PASS')(SKIP,A);                    94
CLOSE_FILES:  CLOSE FILE(MASTER),FILE(NEWMAS),FILE(DETIN),FILE(DETOUT); 95
                                                                        96
```

図2－3　Buchholzのシンセティック・プログラム

```
PRTEND:  GET STRING (END_TIME) EDIT (HRS,MINS,SECS) (2 F(2),F(5,3));          97
   ELAPSED_TIME = HRS*3600 + MINS*60 + SECS;                                  98
   GET STRING (START_TIME) EDIT (HRS,MINS,SECS) (2 F(2),F(5,3));              99
   ELAPSED_TIME = ELAPSED_TIME - HRS*3600 - MINS*60 - SECS;                   100
   PUT EDIT (' ELAPSED TIME = ',ELAPSED_TIME,' SECONDS')                      101
        (SKIP,A,F(7,1),A);                                                    102
   PUT EDIT(' END OF PASS.  REPETITIONS = ',I-1,', SUM = ',SUM,               103
        ', ACTIVE RECORDS = ',COUNT,', CHECK SUM = ',CHECK)                   104
        (SKIP,2(A,F(6),A,F(9)));                                              105
   GO TO START_PASS;                                                          106
                                                                              107
COMPUTE_PASS:  KRETURN = CRETURN;                                             108
   START_TIME = TIME;  GO TO KERNEL;                                          109
CRETURN:  END_TIME = TIME;                                                    110
   GO TO PRTEND;                                                              111
                                                                              112
EOF:  CLOSE FILE(PARAMS);                                                     113
   END YSTKP;                                                                 114
```

Sample data cards

```
                1         2         3         4          CARD
       1234567890123456789012345678901234567890123      COLUMN

        PASS  NMAS=000000 NDET=000000 NREP=010000       COMPUTE PASS.
        PASS  NMAS=001000 NDET=000200 NREP=000001       TAPE PASS.
        PASS  NMAS=001000 NDET=000200 NREP=000050       TAPE PASS.
```

Sample result printout

```
PASS  NMAS=000000 NDET=000000 NREP=010000  COMPUTE PASS.
ELAPSED TIME =    21.2 SECONDS
END OF PASS.  REPETITIONS =  10000, SUM = 4015, ACTIVE RECORDS =   0, CHECK SUM =      0

PASS  NMAS=001000 NDET=000200 NREP=000001  TAPE PASS.
ELAPSED TIME =    16.8 SECONDS
END OF PASS.  REPETITIONS =      1, SUM = 4015, ACTIVE RECORDS = 200, CHECK SUM = 100500

PASS  NMAS=001000 NDET=000200 NREP=000050  TAPE PASS.
ELAPSED TIME =    22.7 SECONDS
END OF PASS.  REPETITIONS =     50, SUM = 4015, ACTIVE RECORDS = 200, CHECK SUM = 100500
```

図2－3の続き

その他，David 等はIBM－360 シリーズのいくつかの機器構成について，シンセティック・プログラムによるJob Streamを与え，パフォーマンスの評価を行った例を報告している。

David 等の採用したシンセティック・プログラムもBuchholz のそれにほぼ近いものである。

プログラムは *PL/I* で記述され，その処理過程は図2－4の様になっている。

(a) マスタ・レコードのデータ・セットを作成する。

(b) このデータ・セットのレコードを処理する。

(c) 処理される各々のレコードに対して，演算のカーネルを多数回実行する。

(d) 各々のレコードが処理されたあと1個から3個までのデータ・セットに出力する。

(e) 各々のレコードが処理された後で数行印刷する。

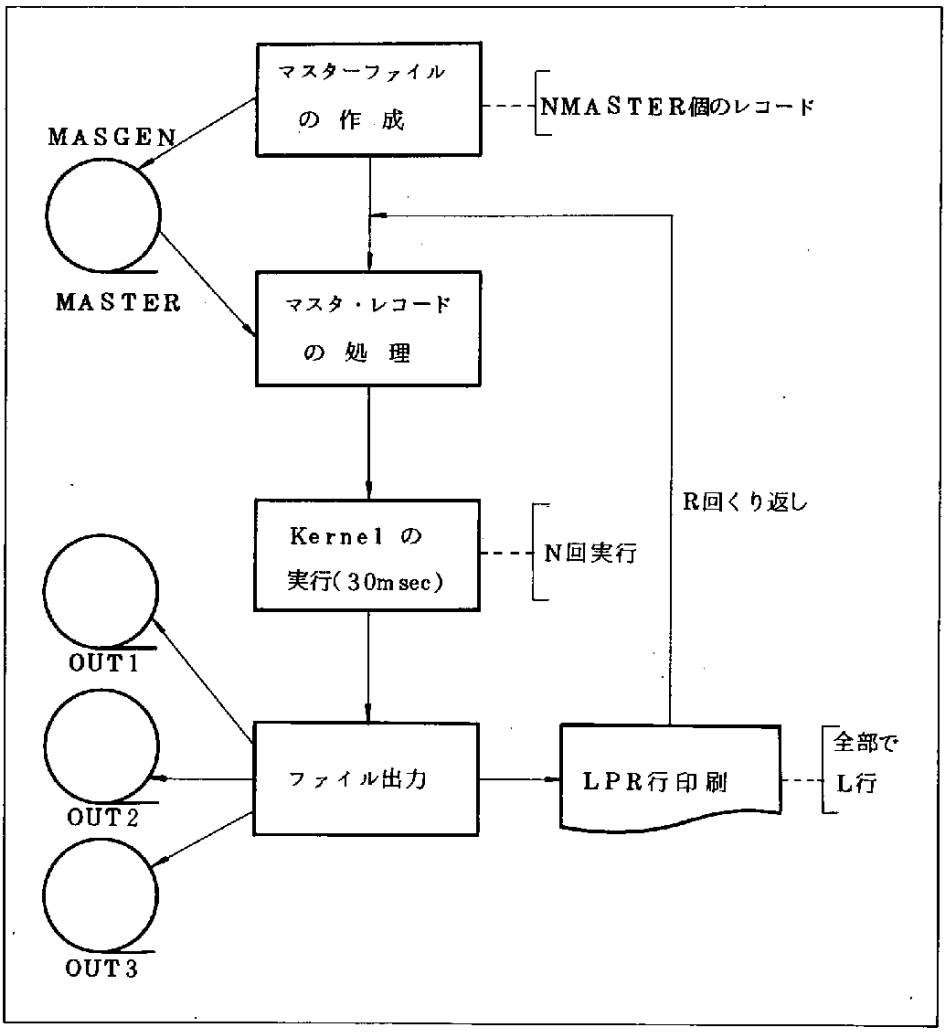


図2－4 Davidの用いたシンセィック・プログラム

ここで，マスタ・レコードの数，マスタ・ファイルから処理されるレコードの数，コンピュート・カーネルのくり返し回数，全体の出力印刷行数，各レコードが処理される時に出力される印刷行数等はすべてパラメータとして与えることが出来る。又Job Mix を作成するに先立ってその特質を知る目的からアカウンティング・ルーティン（ SMF ），ハードウェア・モニタ（DYNAPROBE ）等を使用して実際の Job ストリームに於けるリソースの使用状況についての測定を行ったと云う。

とりあげた項目は次の通りである。

コア占有量， CPU 利用率，チャネル使用時間，周辺装置利用率

さらに，興味深い点はこれ等の情報をもとにして作成したシンセティックJobが果して妥当であるかどうかを調べるために実際のJob Mix とそれに対応するシンセティック・プログラムによるJob Mix とを計算機で処理し，その結果についての検討も行っていることである。表2−17は，その結果の一部を掲げたものである。

表2−17 Activity Monitored with Hardware Monitor
(Minutes)

	Synthetic	Representative
CPU Active	32.4	34.2
Supervisor (OS/HASP)Time	22.7	19.0
Selector Channel 1 Busy	12.4	20.0
Selector Channel 2 Busy	13.5	30.4
Selector Channel 4 Busy	14.6	35.1

その他，前述したCONTEST等もシンセティック・プログラムの範中に含めてよいだろう。

CONTESTはFORTRAN の入出力オペレーションの効率を測定するために考案された一種のテスト・プログラム・ジェネレータである。

必要とする入出力分布のパターンを記号言語で記述し CONTESTに与え

ると，それに対応するプログラムが生成されると云う仕組になっている。
又生成されたプログラムに対してパラメータを与え重みの変更を行うと云うことも可能になっている。

以下にCONTESTによってテスト・プログラムの生成を行った例を示す。

表2－18　CONTESTによってテスト・プログラムの生成を行った例

```
    ( 30 * C - 40 * R( 5 ) - 20 * W( 6 ) )
```

これは，まず，算術代入文を30回実行し，次に，装置番号5のREAD文を40回実行し，最後に，装置番号6のWRITE文を20回実行する入出力文使用パタンをあらわしている。CONTESTは，これに対し，次のようなプログラムを生成する。

```
      SUBROUTINE CNTST
C     (30*C-40*R(5)-20*W(6))
      DO    10  I9999=1,30
      A=((9.0+8.0)*7.0/3.0)**6
   10 CONTINUE
      DO    20  I9998=1,40
      READ(5,1) TIKEA
   20 CONTINUE
      DO    30  I9997=1,20
      WRITE(6,2)
   30 CONTINUE
      RETURN
    1 FORMAT( 8F10,0)
    2 FORMAT(21H *** CONTEST TEST ***)
      END
```

F. シミュレーション

計算機システムの総合評価の道具として，シミュレーションは現在最も有力な手段とされている。現にこれまでにも，リアルタイム・システムやオンライン・システムの設計評価において，非常に重要な役割を果してきた。シミュレーションの大きな利点はモデル化にかなりの柔軟性を備えており，対象とする系に適合する自由度が高いこと，またモデルのきめ細かい記述が可能であり，かなり詳細な解析が出来るという点である。このような理由から，現在，システム総合能力の評価，OS制御アルゴリズムの検討，システム隘路の検出等において最も有力な解析手段となっている。これはシステム評価の目的から見ても，機種の選択，パフォーマンスの改善，システムの設計のいずれにも適合するものである。

シミュレーションとは，実際の系と等価な動作をするモデルを計算機内に作成し，そのモデルを用いて複雑な系の振舞いを解析しようとする方法である。

現実のまねごとという意味で，後述する解析的手法も広義のシミュレーションに含めることが出来るかもしれない。しかし一般には，モデルの性格の違いから，解析手法によるモデルはアナリティック・モデルと呼び，ここで扱うフィジカル・モデルと区別している。

アナリティック・モデルは，システムを待行列理論等に基く数式モデルとして表現したものであるのに対し，フィジカル・モデルはシステムの実際の動作をモデル化したものである。即ち事象の起る事刻表（スケジューリング・テーブル）を保持していて，一般に確率分布を用いて評価しようとするシステムの各要素の振舞いを記述する。このタイプのシミュレーションを事象追跡型シミュレーション（event oriented simulation）と呼んでいる。

両者を比較すると，アナリティック・モデルでは数学的に解析可能な範囲が限られており，複雑化したシステムは扱えなかったり，モデルの表現がど

うしても粗いものになったりするのに対し，フィジカル・モデルではかなりのディテールにわたって対象とする系を記述することが可能であり，モデルが少々複雑化しても十分扱い得るという利点がある。また，前者がモデル化に多くの数学的仮定を要するのに対し，後者ではそれほどそれを必要としないという点もあげられるだろう。

モデルの記述には，FORTRAN等の汎用言語が用いられることもあるが，一般には，GPSS，SIMSCRIPT，SIMULA，SOL 等のシミュレーション言語が多く使われている。

GPSS では，システムの中を動きまわる実体としてトランザクションがある。システムは，その動作を表わすブロック網の中をトランザクションが移動することによってシミュレートされることになる。

モデルの記述に関しては，計算機システムへの処理要求やメッセージがTRANSACTIONに，また各種システム・リソースがＦＡＣＩＬＩＴＹやSTORAGE というふうに直接的な対応があり，そのモデル化の概念は計算機システムのマクロな動作表現には比較的適しているといえる。しかし，モデルが複雑化すると，その細部を記述するのに，かなりもってまわった表現をしなくてはならないとか，TRANSACTIONの流れが明らかでなくなるといった欠点が指摘されている。

ＳIMSCRIPT ではシステムを状態と事象で表現する。

即ちシステムの動作は，事象が発生した時にとられる動作を対応づけるという形で記述される。

GPSS に比べると柔軟性のあるモデル表現が可能だが，システムの動きを事象の生起に分解する作業が大変であり，記述性に問題があるとされている。

以上の様な事情から計算機システムをシミュレートするための専用言語も一部で開発されている。

この点については後に問題点の項でふれることにする。

シミュレーションによってシステムの解析を行った例はこれまでにも数多

くの報告がある。

これを目的別にみるなら，大体次のように大別されよう。

① ハードウェアの設計，評価に重点を置いたもの。

② OSの設計，評価に重点を置いたもの。

③ 総合システムの性能評価に重点を置いたもの。

①は，新しいハードウェアを設計する時，その効果を予測するため主にハードウェア設計者の立場から実施されるものである。

②は，新たにOSを設計する時の改良の指針を定めたり，その効果を検証したりする時，また現行のシステムのシステム隘路の検出を行ったりする目的で，OS設計者の立場から実施されることが多い。

③は，特に利用者の立場からシステムの選択を行ったり，最適機器構成を決定したりする時に実施される。

実例をあげるなら①に関しては，例えば電総研におけるETL MARKVIVに関するGPSSによるシミュレーション実験の結果が報告されている。

また，キャッシュ・メモリのメモリ・ハイアラキーに関して各種の構成，動作アルゴリズム等について比較，検討を行った例も数多く報告されている。

例えば京大の北川等はFACOM230-60の下でアドレス・トレーサを作り，プログラムの動的振舞いに関する検討を行い，その結果を用いてバッファ・メモリの書き出しアルゴリズムを検討するためにシミュレーションを行った。図2－5，図2－6はその結果の一部で，プログラムの動的特性について得られた結果を示したものである。$S(x)$，$B(x)$，$R(x)$ は各々，プログラムの動的特性を表わす量として次のように定義されている。

○プログラムは，どの程度sequencialに実行されるか。

$S(x)=P_r(x=\rho)$：ρステップ sequencial に実行される確率

（$\rho \geq 2$）

○プログラム・シーケンスの変更は，どの程度か。

$B(x)=P_r(x=b)$：bステップ離れた所へブランチする確率

（$b \geq 2$ または $b \leq -1$）

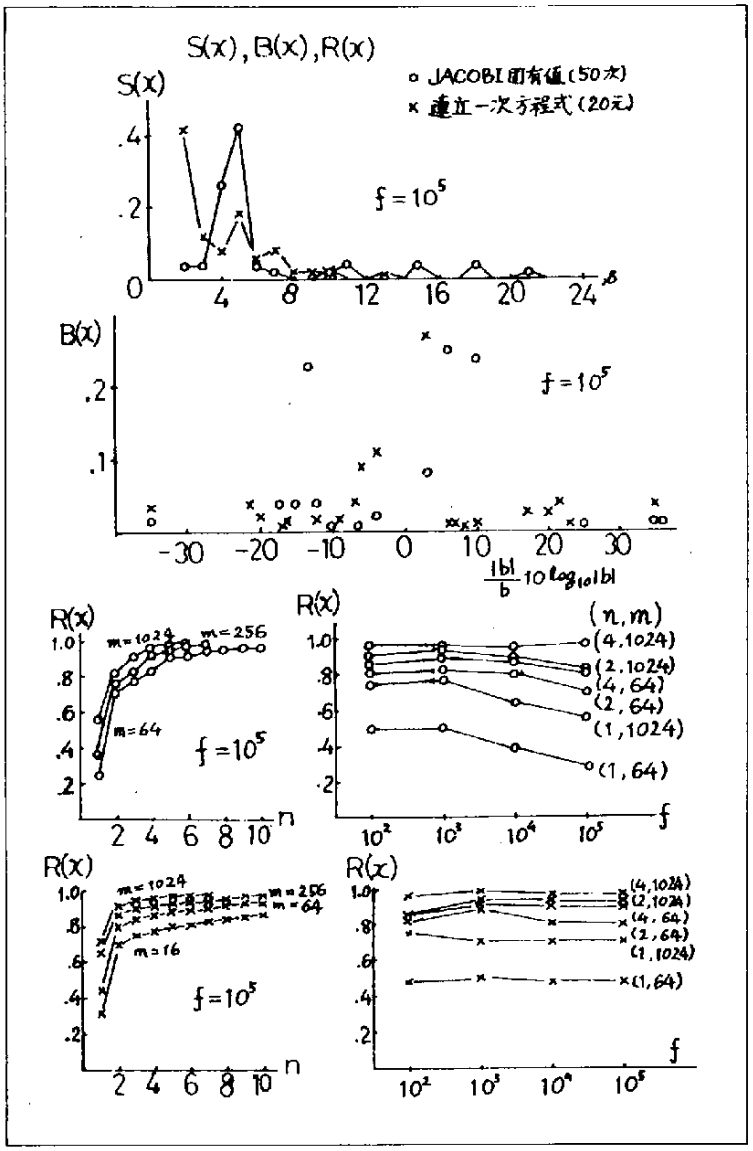


図２－５

○ どの程度，同じ所が参照されるか。

$R(x) = P_r\ (x \leq r)$：プログラムをm語のブロックに分割したとして，あるメモリの参照時にそのアドレスが，最近参照されたr種類のブロックに含まれている確率分布。

○ fはプログラム開始時からのメモリ参照の回数である。

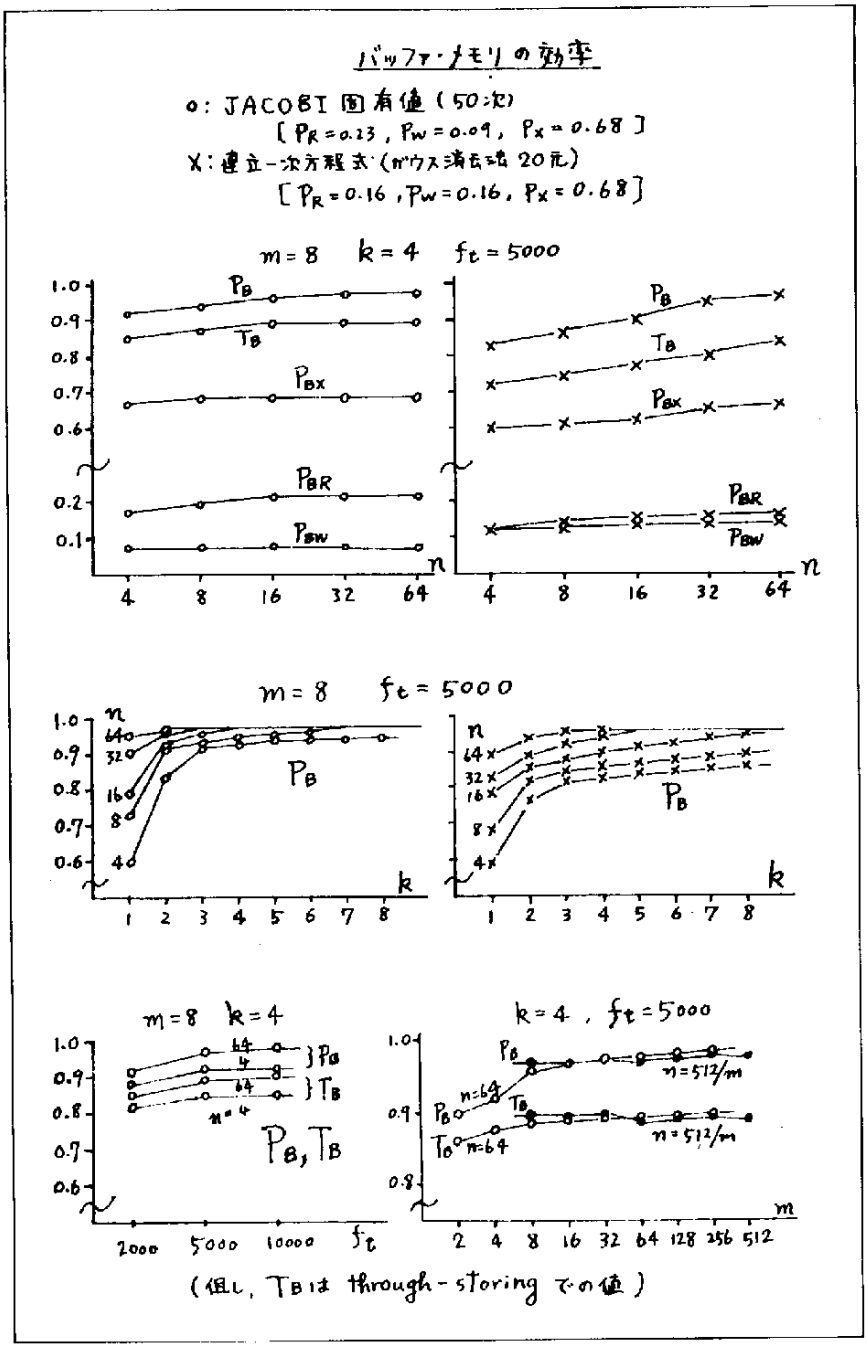


図2－6

f ：CPUからのメモリ参照（命令，データ）の全回数

P_B ：BMUでアクセスした回数／f

P_{BR} ：BMUでデータを読出した回数／f

P_{BW}：BM上にデータを書込んだ回数／f

P_{BX}：BMから命令を読出した回数／f

P_R　：データ読出し回数／f

P_W　：データ書込み回数／f

P_X　：命令読出し回数／f　　である。

書込みアルゴリズムの検討は次の3方式について行っている。

CPUからデータ書込みがあった時

① メイン・メモリだけを更新し，バッファ・メモリ上のそのブロックは無効とする方式。

② 両方を更新するthrough-storing方式。

③ バッファ・メモリだけを更新し，後にそのブロックを消す時にメイン・メモリを更新するpost-storing方式。

図2－6は，書き込みアルゴリズムとバッファ・メモリの効率について得られた結果の一部を示したものである。

次にOSの評価に関するものでは，主にTSSを対象にした研究が多く報告されている。

MITのScherrのCTSSに関する研究は余りに有名である。Fine，McIsaccらはQ32 systemのシミュレーションを行った例を報告している。

また，NielsenはIBM 360/67システム，Merikallio Holland等は航空管制システムのシミュレーションを行った例を報告している。

TSSのシミュレーションに関しては第4章でも述べられるが，ここではN. R. NielsenがB·6500について行ったシミュレーションの例を簡単に紹介する。

N. R. NielsenはB－6500タイムシェアリング・システムに関して，いくつかのタイムシェアリング・テクニックの効果について検討を加えるためにシミュレーション実験を行った。

ここで検討の対象とした項目は次の4つのテクニックである。

① リアロケーション（Realocation）

② クランチング（Krunching）

③ スワッピング（Swapping）

④ ハードウェア・キューア（Hardware queuer）

Krunchingとはセグメント内にある空白の部分を消去してからスワップ・アウトし，スワップ・インした時にはその空白をもとに戻してやる技術を意味する。

また，ハードウェア・キューアとは，ディスク入出力の処理の仕方として，現在のrotational positionと次のrotational positionを比較して，その差が最も小さくなる様な要求を選択してサービスを行うハードウェア機構のことを指している。

ターミナル数は30～45程度を仮定し，バックグラウンド・ジョブも一種のターミナルとして扱っている。また，処理されるジョブとしては36個の標準的なものを用意しているが，これは多くのジョブ統計から引き出されたファイルI／O特性，ターミナルI／O特性，端末ユーザの思考時間等の分布から決定したという。シミュレータはSIMULAで記述され，カードにして約2000枚程度の規模である。

各シミュレーション・ランには，約400枚のデータ・カードが与えられる。データ・カードには，CPUの数とか，ディスクの特性等のシステム構成要素に関する情報がパラメータとして与えられるようになっている。

各シミュレーションに要した時間は約70秒で，最初の10秒間に得られた結果はトランジェントなものとして，統計値に加えない様に考慮している。

表2-19は，その結果の一部（リアロケーションとクランチングの効果）を示したものである。

シミュレーションの結果に関してNielsenは次の様に結論している。

「リアロケーションという技術はタイムシェアリングにおいて非常に有効であるが，その効果がどの程度のものになるかは，主記憶の大きさや，システムの負荷，余ったCPUのサイクルをどの様に利用するかによって左右さ

表2－19

Number of CPUs : 2		
Memory size : 75K worde		
Use of relocation	Yes	Yes
Use of krunching	No	Yes
Percent of CPU time spent		
—in user execution	29.2	34.7
—in system overhead	8.2	18.0
—in idle state	62.6	47.3

れる。

クランチングはスワッピングの能力に比べて，CPUのサイクルが，かなり余っている場合にだけ有効である。そうでない場合は，クランチング自体が大きな負荷となり，オーバヘッドが増大し意味の無い結果になる。

スワッピングによる効果は，システムの全体的なバランスによって左右される。スワッピングの能力に限界がある場合には高速の装置を使用するのが非常に有効である。しかし余力がある時には改善しても，必ず効率が良くなるとは言えない。それは他の部分がネックとなるからである。

ハードウェア・キューアはディスクに対する要求が高い場合には非常に有効であり，大低の場合この機構はシステム・パフォーマンスの向上に役立つであろう。」

また，最後にNielson は次の様に述べているが，これは留意に値することであろう。

「ある特別なテクニックの効果は，システムの置かれている環境（Operating environment），たとえばシステムの設計，ワークロード，2次記憶装置の能力等の各種の変数の関数としてとらえなければならない。」

次に，システムを総合評価するシミュレーションの例としては，SCERT，CASE 等の商用シミュレータが有名である。特にSCERTは1962年，米

国のCOMRESS社によって開発されて以来6回の改訂を重ね，現在，米国，欧州に400以上のユーザを持っているといわれている。SCERTは数学的なモデル作成手法と，経験データを集めたファクタ・ライブラリを組み合わせることにより，各種のシステムに対してパフォーマンスの予測を可能にしたもので，この種のシミュレータとしては最大規模のものとなっている。CASE（Tesdata社）もSCERTと良く似た性格を備えている。その他の商用シミュレータとしてはAMAP（IBM社），CSS（IBM社），SAM（ADR社）などがある。これらの詳細については，第3章に述べられているのでその項を参照されたい。

以上に述べてきたように，シミュレーションはシステム評価の有効な手段となっている。しかし，シミュレーションにも全く問題がないわけではなく，今後ともとり組まなくてはならない問題を数多く残している。よく問題にされるものにコストの問題とモデルの妥当性（Validity）の問題がある。その他にも，シミュレーションそれ自体の問題に含まれるかも知れないが，まだいくつかある。ここでは特にそれらの問題点をシステム評価という立場から取りあげ，その解決の方向とそれに関する研究のいくさつの概略を述べたい。

① コスト高の問題

コストの問題はシミュレータの開発コストの問題と，シミュレーションの実行そのものに費される費用の問題との2つの側面から検討を加えなくてはならないだろう。

モデルが複雑化すると，シミュレーション実験に先だってシミュレータそのものを開発する労力と時間は馬鹿にならないものになる。よくシミュレータが完成した時にはもう必要が無くなってしまったという笑い話があるが，実際その様な例が少なくないということだろう。

久保（日電）は，シミュレーションを実施するまでの各段階で必要とされる費用と，時間について図2－7の様な結果を報告している。

この例からも明らかな様にシミュレータの作成段階までに総費用の約半

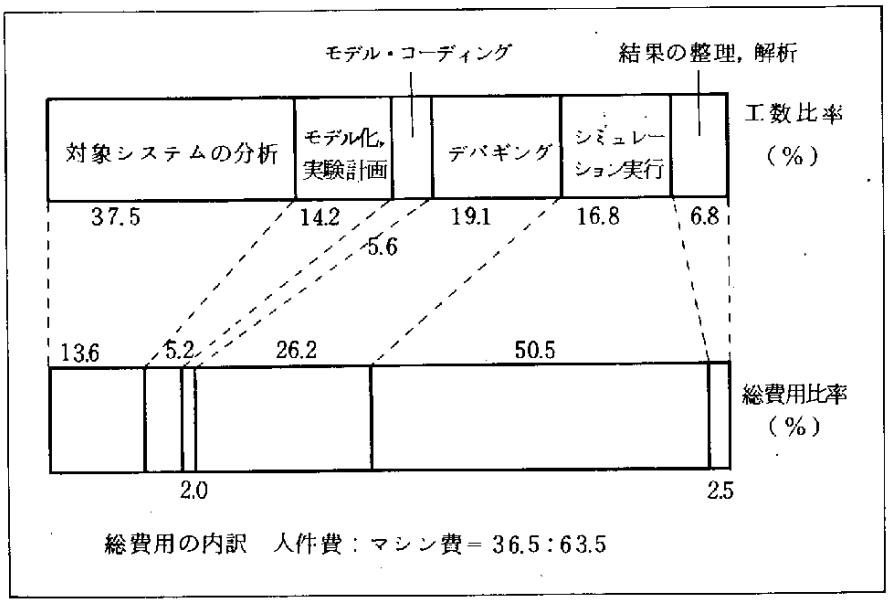


図２－７　大型シミュレーション評価のコスト内訳例

分が費されていることが分る。

シミュレーションの実行の効率化と共に開発工程の効率化がいかに大事であるかが分る。

開発工程の効率化を計るためにまず考えられることは，効果的なモデル作成，デバッギングを可能にするような言語の開発であろう。

この様な目的から計算機システムの専用シミュレーション言語として，すでにいくつかの言語が開発されている。例えばＩＢＭのＣＳＳ，日電のＰＡＣＣＳ，鉄研のＳＰＳ等はよく知られているものである。

実行段階の効率化に関しては，まずシミュレーション・ランそのものに時間を費さない様な努力が必要である。例えば，時間がかからないようにモデル化を進める等が考えられる。もう一つはシミュレーションの試行錯誤過程の計画的実施であろう。これには実験計画法その他の手法をとり入れて最適化を計る必要がある。この点に関しては，後にもう少し詳細を述べる。

その他，シミュレータのオンライン化をはかる事も一つの有力な手段と

なろう。

② モデルのValidityに関する問題

現在のところ，モデルの妥当性をチェックする適当な手段はない。多数回のシミュレーションの実施と，それに対する実測データに基く結果の検証というサイクルの中でモデルを洗練し多くの経験的事実を積み重ねてゆくより仕方ないであろう。また，対象とするシステムそのものの構造を別の面から分析しそれを反映してゆくということも大切である。

いずれにしろヒューリスティックな過程を経て今後解決してゆかなくてはならない問題の一つである。

③ ワークロードの問題

Nielson も述べている様に，ワークロードはそれ自体，システム・パフォーマンスに影響を与える一つのファクタとしてとらえなければいけない。

例えばある政策の実施効果を見ようとするとき，各々異った性格を備えたワークロードの下で比較すれば，結果は当然異ったものとなり，政策の実施が果して効果があるかという点に関してとまどうことになる。

この様にワークロードの性格を明らかにするということはシステムを分析する上で重要な問題の一つとなっている。

ワークロードの性格を規定するのには，一般に各種リソース（メモリ，チャネルetc）に対する要求の分布という形で行なわれる。

シミュレーションにおいては，これらの分布に適当な仮定をして，それをモデルに組込むという形をとるわけであるが，果してそれが妥当かどうかという点に関しては，十分に保証されていないのが普通である。

こうした事情から，実際のワークロードに関するデータを収集しその結果を用いてシミュレーションを行おうとする試みもある。これはトレース・ドリブン型（trace driven type）シミュレーションと呼ばれている。

例えば日立の益田等は，HITAC 5020 TSSの下でPATTERN と呼

ばれるアドレス・トレーサを開発した。

PATTERNは指示により，ユーザ・プログラム，OS，セグメントの各単位についてアドレス軌跡をとることが可能であり，結果はテープに書き出される様になっている。実際にPATTERNを使用して，4種類のプログラムにつきアドレス・トレースを行い，その結果を用いてスワッピング・アルゴリズム，主メモリ・サイズ，ページ・サイズを検討するためシミュレーションを行ったことを報告している。

次の表2-20は，その結果の一部を示したもので，テーブルは測定対象となったプログラムに関する記述である。

図2－8はそのアドレス・トレース・データを用いて行ったシミュレーションの結果である。

4つのスワッピング・アルゴリズム，LRU，FIFO，RANDOM，OPTIMUMについて主記憶サイズと次のページ・フォールトが起るまでに走るプログラムの平均走行距離（msbf. Mean Step Between Page Fault）の関係を求めている。スワッピング・アルゴリズムについては次の様に説明されている。

表2－20　Programs to be analyzed.

	page size (word)	program size (page)			number of segments used			program steps (on Mag. Tape)
		proc.	data	sum	proc.	data	sum	
BASIC	64	44	547	591	5	8	13	549,105
	256	15	143	158				
	1,024	7	43	50				
PL/1W	64	148	194	342	12	19	31	356,173
	256	47	70	117				
	1,024	18	38	56				
CONCISE	64	56	67	123	6	7	13	514,972
	256	16	25	41				
	1,024	8	12	20				
EIGEN	64	16	171	187	5	5	10	195,296
	256	6	47	53				
	1,024	5	16	21				

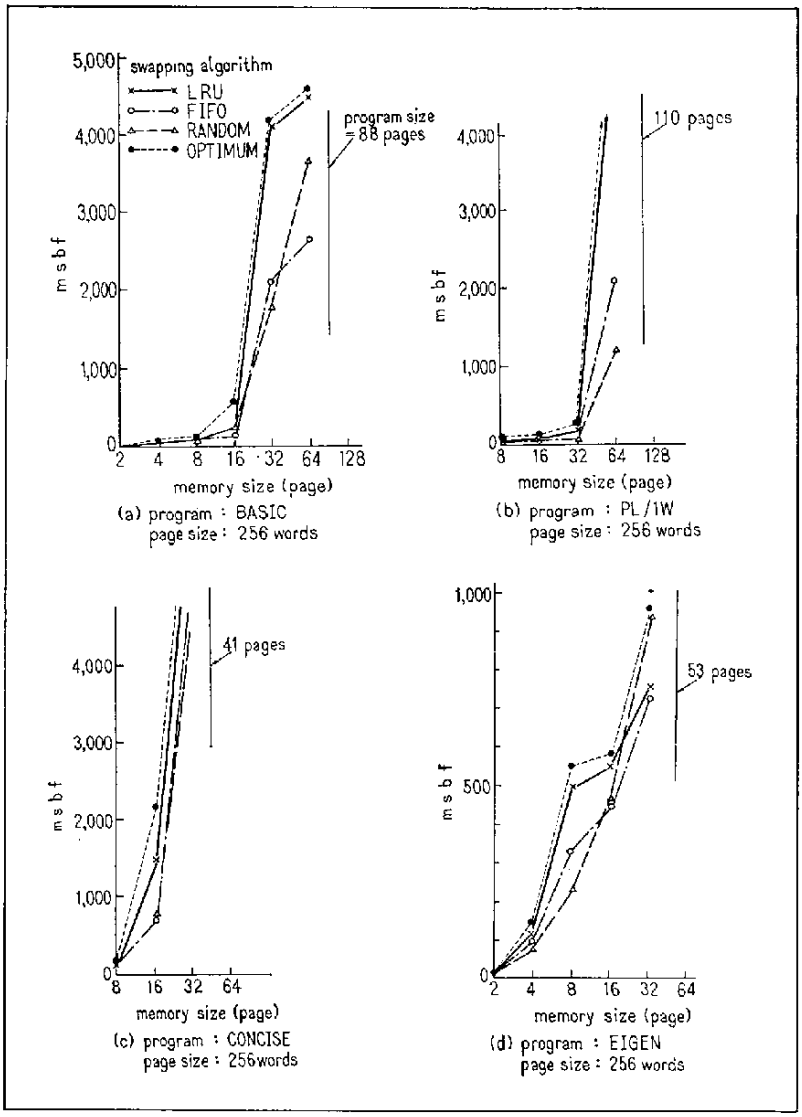


図2－8　msbf（mean step between page fault）vs. memory size, page size and swapping algorithm.

LRU　：（Least Recently Used）スワップ・アウトの必要が生じた時，時間的に最も遠い過去に参照されたページをスワップ・アウトする。

FIFO　：主記憶にあるページのうち時間的に最初に読み込まれたペ

ージをスワップ・アウトする。

RANDOM：主記憶内のページをランダムに選択してスワップ・アウトする。

OPTIMUM：スワップ・アウトする必要のある時点から将来をながめた時，現在メモリー内に存在するページのうち最も遠い将来まで参照されないページをスワップ・アウトする。

又，アドレス・トレースに関連してこれをハード的に収集する方式としてUNIVAC 1108 マルチ・プロセッサー用に開発されたデータ収集装置が知られている。

この他にもプログラム動作の分析と云う面での研究はかなり行われている。

最も基本的なものから考えると命令の使用頻度の分析があげられる。

又，ページング・システムで無いシステムに関してはSVC の系列 とSVC と SVC の間のプログラムの走行時間等が分析の対象となっている。

これはOSのどの部分の使用頻度が高いかとか，スケジューリング・アルゴリズムを検討する上で貴重な資料となる。

ページング・システムに関してはページのリプレーシング・アルゴリズム等を検討する目的から，新しいページの要求を時系列としてとらえようとする様な試みが多く為されている。

例えば図2－9はFineがSDC Q－32 TSS のもとでプログラムのアドレストレースを行い，経過時間とページ・デマンドの関係について得た結果である。

これに類する研究も数多い。

又，Denningはプログラム動作に関する一般的なモデルとしてワーキング・セット（Working Set）と云う概念を導入した。Working Set，$W(t,\tau)$は時刻 $t-\tau$ から時刻 t までの間にそのプログラムが参照したページの集合として定義される。現在，ワーキング・セットをOSのスケジューリング・アルゴリズムの中にとり入れて利用しようとする動きもある。

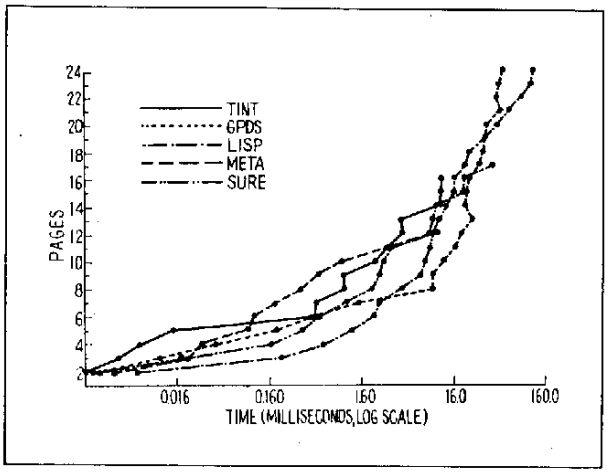


図２－９　Page demand（SDC）

その他，ＴＳＳやオンライン・システムで問題になる端末ユーザの振舞いに関してその行動を分析しようとする様な研究も行われている。

例えば米国ＮＢＳではＤＩＡＬＯＧＵＥ　ＭＯＮＩＴＯＲ と呼ばれるモニタをターミナルとホスト・コンピュータの間に置いて，端末ユーザの振舞いを記録する機構を実現し，端末ユーザの行動に関する分析を行っていると云う。

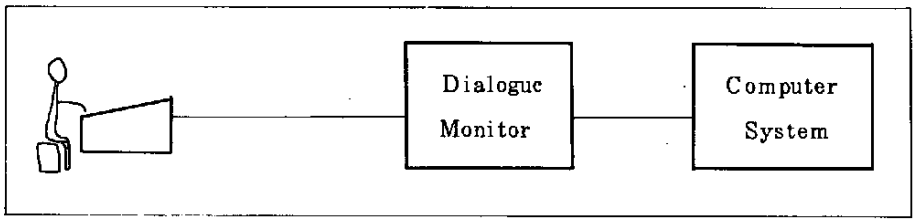


図２－１０　米国ＮＢＳでの実験

④　効率と精度の問題

シミュレーションに於ける重要な問題として，シミュレーションに必要とされる時間と精度の問題がある。

これには３つの側面から検討を加える必要がある。

一つはモデル作成段階に於ける考慮である。

即わち，精度を上げるためにモデルを精密化すれば実行に要する計算時間が増大し；又モデルが粗すぎると時間は短縮されるが精度が落ると云ったジレンマをどのように解消するかを考えなくてはいけない。

これにはまずモデルを簡略化するための各種の仮定を結果の精度が及ばない範囲で設定することが重要である。

そのためにはシミュレーションの予備的段階に於ける対象システムの分析が重要である。

又得られた結果を実測データをもとに検討し，それをモデルにフィードバックすると云うことも考えられよう。

その他粗いモデルと組み合わせて検討する方法も効果があるだろう。

もう一つの問題は一回のシミュレーションに要する時間をどこで打ち切るかと云う問題である。

いたずらに長くシミュレーションを続けても意味が無いであろうし，又短か過ぎると定常解が得られないで終ってしまうと云う結果になる。

適当な収集判定条件の設定により，精度と時間をバランスさせることが重要になる。

その他，最適パラメータの決定過程では全シミュレーション・ランに要する時間の短縮をはからなくてはならない。

これには，Direct Search法，Slide法，シミュレータの学習機能と云った各種手法が提案されている。

（金沢，北川他「計算機システムのシミュレーションとその効率について」第13回 プログラミング・シンポジウムを参照されたい）

G. モニタリング

モニタリングとは実際に稼働しているシステムを観測し，そこからパフォーマンスに関するデータを収集する手法を云う。これはシステムの振舞いに関する実際のデータが直接的に得られると云う意味でシステムを分析，評価する上での有力な手段となっている。

測定データを得る手段としては最も手軽な方法として，アカウンティング情報を利用することが考えられる。

しかし，ここから得られる情報は比較的マクロなレベルのものだけであり，システムの細部の振舞いを把握するには向かない。

この様な目的に沿ったものとして一般にはハードウェア・モニタ，ソフトウェア・モニタがよく使用されている。

ハードウェア・モニタは計算機から直接，電気的信号を検出しこれを収録する方法である。

これはシステムに対し測定による外乱を与えないという意味で優れている。

しかしプローブと測定点との接続は素人向きではないし，又得られた情報に記述性が無いためデータ加工に注意を要する等の欠点がある。

一方，ソフトウェア・モニタはOSの内部又は外部に組み込まれた監視用のソフトウェアにより必要な情報を収録する方法である。

前者に比較すると，これは測定がシステム動作に影響を及ぼすと云う点で使い方に注意を要する。

しかしシステム内のソフトウェアと関連性のとれたデータが収録できると云う点では優れている。

従って実際の利用にあたっては以上述べたような各々の特徴をよくふまえた上で目的に応じた使い分けをすることが肝心である。

モニタリングのシステム評価に於ける利用範囲を考えると大体次の様になるであろう。

① システムの改善

OS の隘路を検出したりプログラムの振舞いを分析したりすることがで

きる。

② 最適機器構成の決定

各種リソースのユティリゼーションを分析しながら最適機器構成を決定してゆく。

モニタリングによるシステム評価は，データの収集と結果の整理と云う2つの段階を経て行われる。

データの収集段階に於ては，測定がパフォーマンスに与える影響を少くする事と測定対象としてどの様な量を選ぶかについて十分考慮しなくてはいけない。

ハードウェア・モニアの場合は測定上の外乱はそれほど問題にならないかもしれないが，ソフトウェア・モニタの場合には避けられないものとなってくる。

この場合，あらかじめ測定による影響が測定値に対してどの程度及ぶものかを検討しておき，結果の精度を十分認識する必要があるだろう。

又評価量としてどの様な量を測定対象として選ぶかと云う事も重要である。

この点についてはまだ完全に整理されていないのが現状である。

いくつかの測定例をもとにして，よく測定対象とされるものについて整理すると大体次の様になる。

① ハードウェア特性を測定する目的から

- 各命令の使用頻度
- 平均命令実行時間

② OS の特性を分析し改善する目的から

- SVC の時系列
- SVCとSVCの間のプログラムの走行距離の分布
- OS 各モジュールのアクセス頻度
- ワーキング・セット（Working Set）

③ ユーザ・プログラムを改善する目的から

- プログラム各部の使用頻度

④　システムを改善する目的から

・各種リソースのユティリゼーション

⑤　Job Mix を分析する目的から

・Job統計

又測定の結果得られた多量のデータをどう処理し，最終的にシステム評価とどう結びつけるかと云う事も大切である。多くの場合は得られた情報そのものがシステムを評価する直接的な尺度とならないのが普通である。

その結果に対して何等かの統計的処理（例えば回帰分析等）を加えるとか，その結果を用いてシミュレーションを行うと云ったことが必要になってくるであろう。

以下，ハードウェア・モニタ，ソフトウェア・モニタについてもう少し詳細な説明を加え実施例のいくつかを紹介する。

(1)　ハードウェア・モニタ

ハードウェア・モニタはプローブと呼ばれる検出端をシステムと電気的に結合することによりシステムの動作に関する状態信号を収集，記録する装置である。

その一般的な構成は図 2－11 の様になっている。

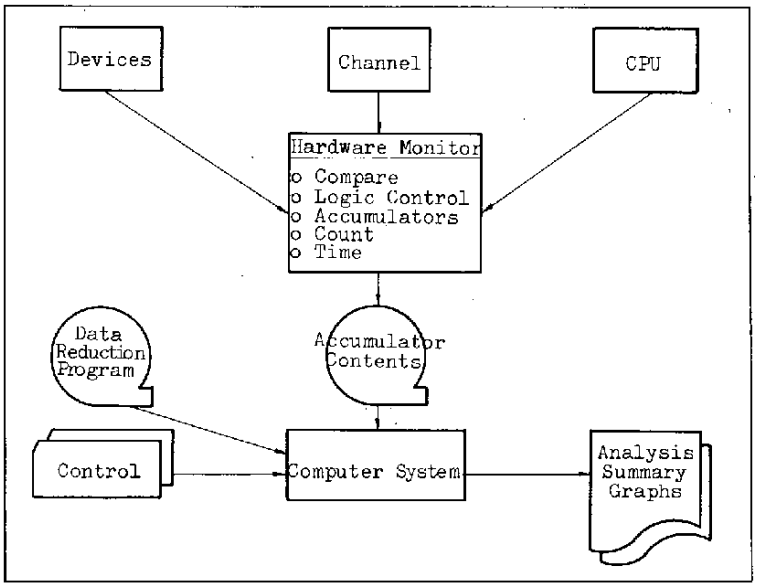


図 2－11　一般的なハードウェア・モニタの構成

・プローブ：必要とする情報をとり出す検出端。接続位置（ピンポイントと呼ぶ）はシステムによって固有である。従ってプローブの接続にはCEとの共同作業が必要になる。一般に，商用のものではピンポイントに関するマニュアルがサポートされているのが普通である。

・ロジックボード：とり出した信号を適当に合成して意味のある信号に変換するためのロジックが附属している。

・タイマー：サンプリング・レートを設定する。

・カウンター：とり出された信号（パルス）をカウントする。

・磁気テープ：とり出した情報を記録する。

その他，得られたデータを加工し，レポートを作成するソフトウェアも附随しているものが多い。

ハードウェア・モニタは独自に作成することも可能であるが，製品化されて市場に出まわっているものもいくつかある。

その他，計算システムそのものがモニタリング機構を備えている様な場合もある。

例えばIBM 360/67 ではシステムのパフォーマンスを変えずに仮想記憶装置から主記憶装置へのスーパバイザの移行，共有プログラムとリエントラント・プログラムの動き，チャネルの利用率等が監視できるようになっている。特別に作成した例では，例えばSYSDAS（日電），CPA（富士通），SMI（IBM）等が報告されている。

又，モニタリングに小型計算機を利用した特殊な例としてはUCLA で実験されたSNUPERが報告されている。商用ハードウェア・モニタの例ではCOMRESS 社で開発されたDYNA PROBEが有名である。

その他XRAY（TESDATA社），SUM（Computer Syne ctie 社），ME（B&B社）等が知られている。

これらの詳細に関しては第3章にゆずるとしてここではSYSDASによる測定例を簡単に紹介する。

日電・箱崎等はハードウェア・モニタSYSDASを開発しシステムの測定を行った例を報告している。

図2−12はSYSDASの構成を示したものである。構成上の特徴は連想記憶を採用することにより融通性のあるデータ収集を可能にした点である。

測定はハードウェア特性，プログラム特性，システム特性についてそれぞれ行っている。

図2−13〜図2−16はその結果の一部を示したものである。

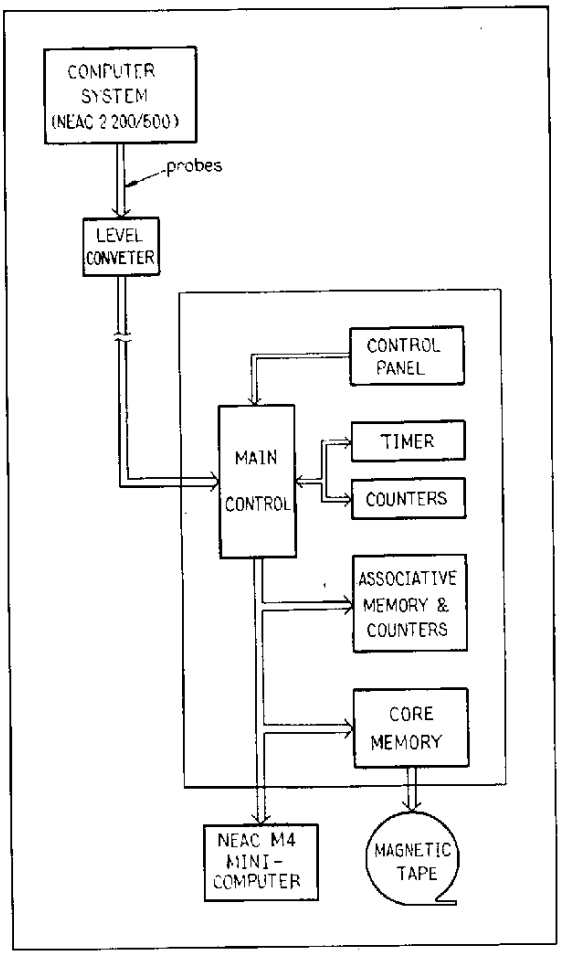


図2− 12　SYSDASの構成

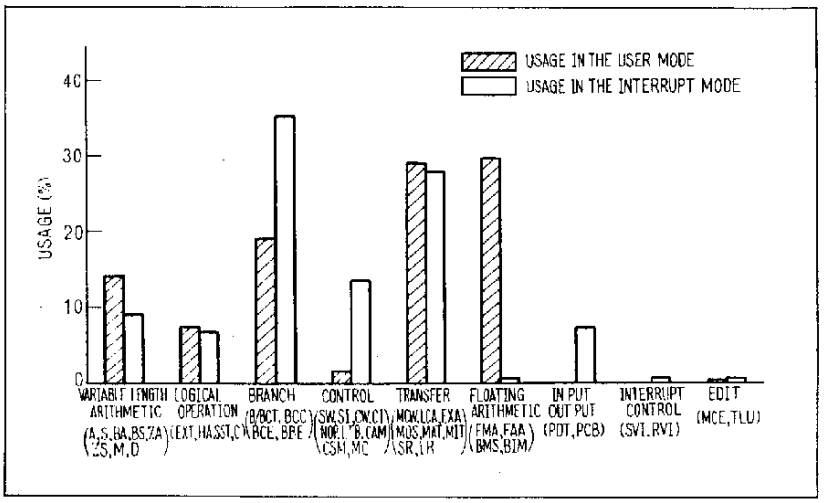


図 2 − 13 Distribution of instruction usage

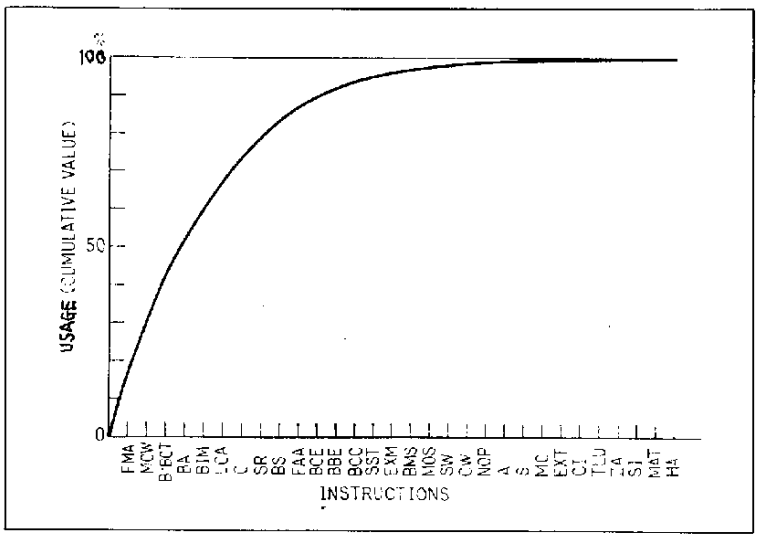


図 2 − 14 Instruction usage in the user mode

PROG. SIZE		PROGRAM TYPE	SVC INTER-VAL (MEAN)
A	12 K-CHAR.	INPUT READER	210 Steps
B	200K	FORTRAN COMPILER	4,970
C	200K	PROBLEM PROGRAM (FORTRAN)	16,820
D	60K	PROBLEM PROGRAM (FORTRAN)	1,823
E	—	PROBLEM PROGRAM (BUSINESS USE)	500

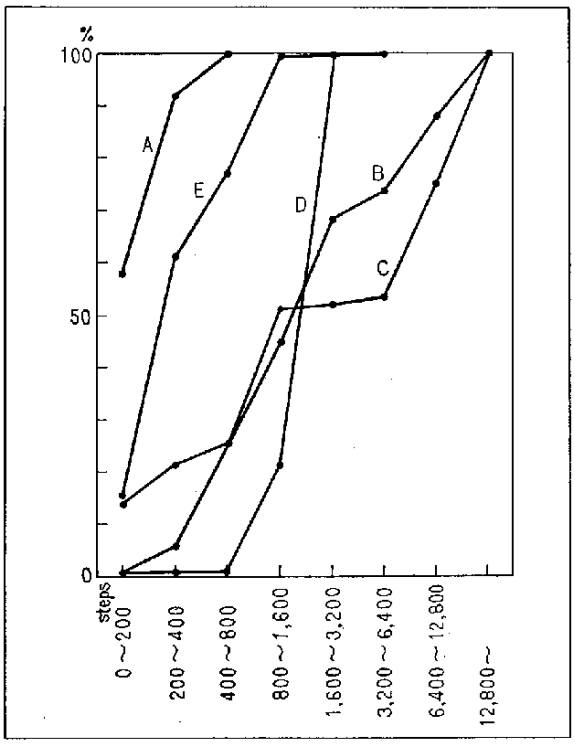


図 2 − 15 Cummulative distribution of SVC interval steps

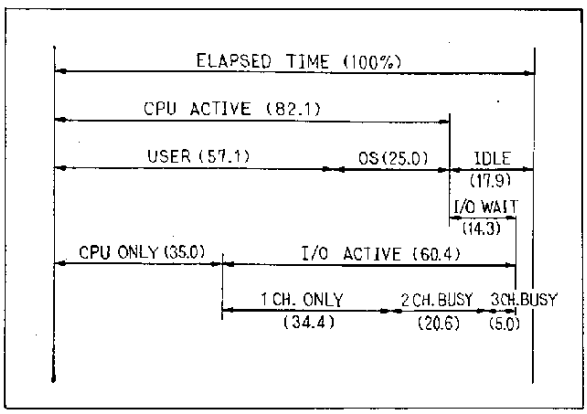


図 2 − 16 System profile of a measured system

(2) ソフトウェア・モニタ

ソフトウェア・モニタとはシステム内に組み込まれた，測定用プログラムにより，システムの振舞いに関するデータを収集する機構を云う。

一般にその動作原理からイベント・タイプとサンプリング・タイプに分類される。

イベント・タイプのものは一般にOSの核であるタスク管理部に組み込まれていてイベントの発生毎にOS内部にあるシステムの状態に関する情報を外部にはき出す形式をとる。

一方，サンプリング・タイプのものはある一定時間間隔《サンプリング周期》毎に必要な情報をとり出し外部に記録する。いずれもOSの内部を参照しなくてはならないからスーパバイザ・モードで動作させるかもしくはメモリ・プロテクションを解くと云った手段が必要であり，ユーザが単独で作成することは仲々困難な場合が多い。又，データ収集の過程はそのままシステム負荷として加わるからそれがシステム動作に与える影響も考慮しなくてはならない。

特にサンプリング・タイプのものではそのサンプリング周期の設定に注意を払う必要がある。

例えば商用モニタとして有名なSMSに関して，Sampling Rate と精度の関係について図2−17のような関係が公表されている。

一方OSを介してデータを得ると云うことは，ある意味で便利なこともある。

それはOSの管理下にあるJobに関してそれに関連ずけられた情報が得られると云うところである。

例えばあるプログラムが現在どの様なリソースを使っていて，どの様な動作状態にあるか等を知ることが出来る。

又サンプリング・タイプのものでは比較的容易にアドレス・トレースを得ることができる。

この様に一般にハードウェア・モニタと較べた時モニタリング・オーバ

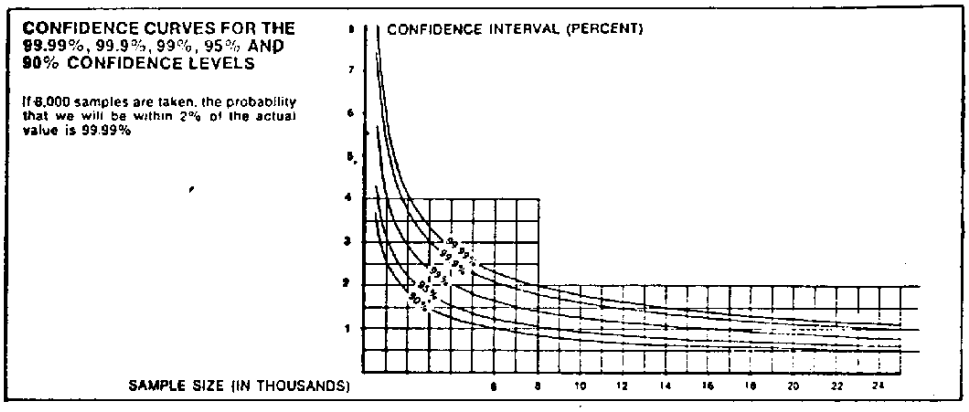


図２－17　CONFIDENCE CURVES

ヘッドは大きくなるが汎用性，融通性，経済性に於て優れているとされている。

ソフトウェア・モニタの場合も測定されたデータは一担外部記憶にはき出され，後にリダクション用のソフトウェアを用いて加工を加えるのが普通である。

特殊なケースとして，小型計算機を連結して収録されたデータを即時に処理しシステムの時々刻々の変化がリアルタイムで観察できるというものもある。（ＳＰＹを参照されたい）

ソフトウェア・モニタは，現在，商用化されたものも含めるとかなり広く利用されているが，独自の目的で開発されたものではＳＩＰＥ（ＩＢＭ），ＡＭＡＰ（ＩＢＭ），ＳＭ（富士通），ＰＭＳ（京大），ＣＯＳＭＯＳ（ＣＤＣ）等が報告されている。

ＳＩＰＥはイベント・タイプに属するものでＴＳＳ／360 の解析用に開発されたと云う。

ＡＭＡＰもやはりＩＢＭ社内の利用を目的として開発されたものでＯＳ／360の下で実行されるユーザ・ジョブの動作データをイベント方式で収集

する。

SMはFACOM 230-60 の下で開発されたものでOSのオプション機能としてシステム・ジェネレーション時にOSに組み込むことができるようになっている。

処理プログラム及びモニタ・モジュールの動きに関するデータをイベント方式で収集するようになっている。

PMS は京大でユーザの立場から開発されたもので，F-60の割り出しの特殊性をうまく利用してOSの管理情報を参照することができるようになっている。タイプはサンプリング・タイプに属する。

商用ソフトウェア・モニタも数多く市場に出ている。有名なものではSMS（B&B），STAGEⅡ（TESDATA），FORMAX（Computer Synectic）がある。

それらの機能，タイプ等については第3章で詳細が述べられるのでこの項では特に説明を加えない。

ソフトウェア・モニタを使用して測定を行った例も数多く報告されているが中でもStaneley，Hartel 等のアポロ軌道制御に於けるリアルタイム・システムのモニタリングの話はよく知られている。

その他リアルタイム・システムのモニタリングに関してはStaneley，Stherr等の報告がある。

又Joseph（CDC）等はUNIVAC 70／7 の下でCOSMOSと呼ばれるソフトウェア・モニタを開発し，ヴァーチャル・メモリ・オペレーティング・システム（VMOS）に関する動作解析を行った例を報告している。

主にThink timeの分布，Compute timeの分布，ページ・フォールトの間隔の分布，入出力要求回数の分布等を対象に分析を行っているがこの詳細については第4章で述べられるのでその項を参照されたい。

H. 解析的手法

解析的手法は計算機システムを数式モデルとして表現し解析を試みる方法である。

モデルは待ち行列理論（Queuing theory）に基くものが多いが，その他OSの構造を有向グラフとして表現するグラフ・モデル等についても研究されている。

主にディスクのパフォーマンスを解析するなどシステム構成要素の一つに注目した局所化された問題を扱う時によく用いられる方法である。

数学的に解析可能な範囲に限りがあるから余り複雑なシステムを綜合的に評価する目的には向かない。

他の方法と比較して有利な点は経済性及びシステム変数間の関係がパラメトリックにとらえられること等にあるとされている。

待ち行列理論は待ち行列を扱う一般のサービス機構を解析するために研究された数学理論で，その先駆的研究は20世紀初頭に始まる。

Danish，Erlang，Pollaczeck，Kolomogorov，Kendall，Takacs等がその基盤を築いたことは周知の通りである。

現在，通信回線網，輸送，一般サービス機構等のサービスを研究する目的で広く利用されているが計算機の分野でもオンライン・システムの設計には早くからとり入れられ利用されている。

一般に計算機システムの解析に於てはポアソン分布に従う入力と一般のサービス時間を持つ単一窓口システムを研究することが最も有効であるとされている。

この時，窓口には例えば，チャネル，CPU，通信回線等の様な呼を処理する任意の装置が対応する。

表2−21はコンピュータ・システムと待ち行列モデルの典型的な対応関係を与えたものである。（W. Changによる）

表2－21　コンピュータ・サブシステムにおける典型的な待ち行列

客	客の発生源	サービス窓口	窓口数	解析の形	待ち行列の形
記憶の参照	CPU；チャネル；記憶	BSUコントロール装置（BCU）	1	BCU	優先度 バルクサービス[1]
記憶の参照	CPU；チャネル；記憶	記憶のモジュール（M）	M	主記憶	複数窓口
システム タスク	I／O装置；CPU；チャネル；記憶；プログラム	CPU	1	マルチ プログラミング	優先度
メッセージ；データ；プログラム	チャネル；CPU；通信回線	バッファ	固定長ブロックのとき1 可変長ブロックのとき変数	バッファリング	複数窓口
チャネル要求	チャネル；CPU	CPU	1	CPU；チャナル インターフェイス	優先度
バイト；データ レコード	I／O装置；CPU；通信回線	チャネル	1	チャネル	競争[2] ；優先度；先着順
I／O要求	CPU；ディスク；主記憶	ディスクおよびチャネル	M＋1	ディスクおよびチャネル	直列待ち行列[3]
端末；プログラム；タスク	端末； I／O装置；CPU；主記憶	CPU	1	タイムシェアリング コンピュータ	フィードバックのある優先度
I／O 要求；ページング要求；タスク	I／O装置	CPUとI／O	2	タイムシェアリング；ページング	フィードバックのある直列待ち行列[4]
記憶の参照；データ レコード	CPU；主記憶；データ・レコード；プログラム	主記憶モジュール（M）と補助記憶装置（N）；高速記憶（M）とキャッシュ（N）	M＋N	階層記憶	フィードバックのある直列待ち行列
入力メッセージ	端 末	通信回線	1	全2重入力通信回線	多重待ち行列[5]
出力メッセージ	CPU	通信回線	1	全2重出力通信回線	先着順；優先度つき待ち行列（ポーリングメッセージが最高優先度）

客	客の発生源	サービス窓口	窓口数	解析の形	待ち行列の形
入出力メッセージ	CPUと端末	通信回線	1	半2重回線	優先度つき待ち行列（出力メッセージが最高優先度）
使用者の端末	集団	使用者の端末	M	工業利用（航空，銀行，デパートなど）	多重窓口

1. バルク　サービス：サービス期間中に複数の客を同時に BCU によってサービスすることができる。
2. 競　争：すべての客がサービスを要求でき，窓口では客をランダムにあるいは，ある順序にしたがってサービスする。
3. 直列待ち行列：1つの待ち行列システムの出力が，ほかの待ち行列システムの入力となるもの（階段状の待ち行列）。
4. フィードバックのある待ち行列：サービスを終った客がさらにサービスを受けるために再び待ち行列に戻るもの。優先度つきの待ち行列システムあるいは階段状の待ち行列システム（直列待ち行列）でおこることがある。
5. 多重待ち行列：ことなる待ち行列に到着した客が同一のシステムによってサービスされるもの。

この様にして設定されたモデルを待ち行列理論に基いて解析的に解くことにより平均待ち時間，平均待ち行列長，遊休時間，サービス時間等を求めることができる。

その結果システムの状態を把握したりいくつかの政策の実施効果を検証したりすることが可能になる。

待ち行列モデルを用いてシステムの分析を行った例は数多くあるが中でも E. G. Coffman, L. Kleinrock, B. Krishnamoorthi, R. C. Wood, N. R. Patel, L. E. Schrage, A. L. Scherr 等の研究は有名である。これ等の研究については第4章で詳細が述べられるのでその項を参照されたい。

図2−18はKleinlock等の扱った，比較的単純化された待ち行列モデル

の一例である。

以下に簡単な説明を加える。

次々に到着する新しいＪｏＢはシステムのキューにつながれる。

その中から一つのＪｏＢを選び出してCPUがサービスを行う。（タイムカンタムを与える）。

もし，与えられたタイム・カンタム内で計算が終了したらそのＪｏＢはシステムを立ち去るが，そうでなければ再びキューにもどると云う過程を表現している。

ここで，解析の便宜上，到着過程はポアソン分布を，又サービス時間は指数分布を仮定している。

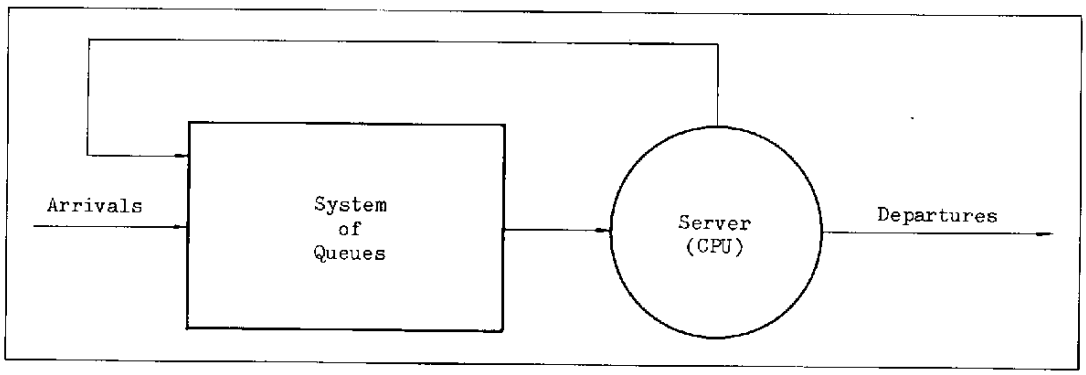


図２－18

検討を行ったサービスのパターンは，次の３つの方式についてである。

① FCFS：first come first served

② RR　：Round Robin

③ FB　：Feed back

次にサービスを行うものは，それまでに最もサービス量の少なかったものが選択される。

図2-19はその結果で，ＪｏＢの要求する，サービス時間tと平均レスポンスタイム$T(t)$との関係を３つの方式について求めたものである。

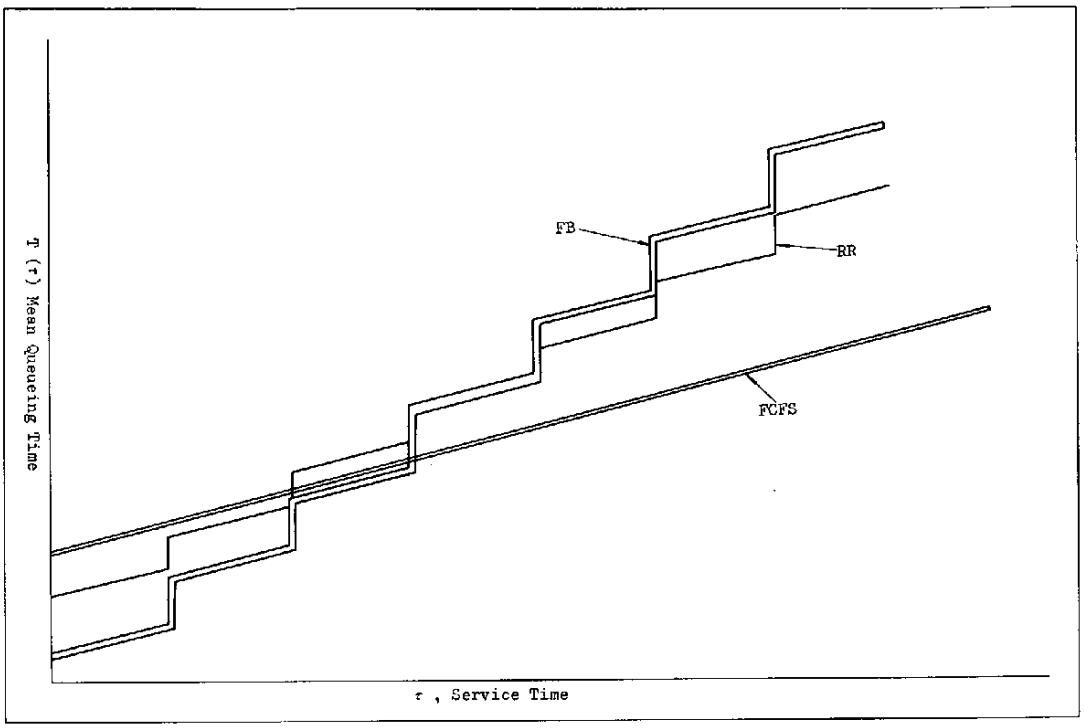


図 2 － 19

参考文献

I Over View

1) R.R. Johnson : Needed : A Measure for Measure, DATAMATION, Vol. 16, NO. 12, 1970, pp. 22−30.

2) H. C. LUCAS, JR : " Performance Evaluation and Monitoring, " Computing Surveys, Vol. 3, No. 3, pp. 79-91 (1971).

3) H. C. LUCAS, JR : " Performance Evaluation and the Management of Information Services, " DATA BASE, Quarterly Newsletter of SIGBDP, Vol. 4, No. 1, pp. 1−8 (1972).

4) H. W. Cantrell and A. L. Ellison : Multiprogramming System Performance Measurement and Analysis, Proc. SJCC 1968, pp. 213−221.

5) Saltzer, J. H. and Cintell, J. W. : The Instrumentation of Multics. Comm. ACM, Vol. 13, No. 8, pp. 495−500, (1970).

6) Drummond, M. E., "A Perspective on Systems Performance Evaluation, " IBM Systems Journal, Vol. 8, No. 4, (1969). pp. 252−263.

7) 萩原 宏 : " 計算機システムの評価について " 情報処理 VoL. 13, No. 11.

II Instruction Mix

8) 石田晴久 : ギブソン・ミックスの起源について, 情報処理, Vol. 13, No. 5, pp. 333−334.

9) K. E. Knight : Changes in Computer Performance, DATAMATION. Vol. 12, No. 9, 1966, pp. 40−54.

10) 井上頼昭 : " ハードウェアの評価 ", 情報処理, Vol. 13, No. 11.

11) 高橋義造 : " コンピュータ評価のための各種ミックス ", 情報処理, Vol. 13, No. 11.

III Benchmark synthetic

12) Arbuckle, R. A. : " Computer analysis and throughput evaluation, " Computers and Automation 15, 1 (Jan. 1966), 12−15, 19.

13) Auerbach Info. Inc. : " Standard EDP Report, " 1971.

14) Buchholz, W. : " A synthetic job for measuring system performance, " IBM Systems J. 8, 4 (1969), 309−318.

15) 高橋 : " 東北大学大型計算機センターにおける OSの効率測定, " 第3回研究セミナー報告, 京大大型計算機センター (1971年12月), 64−70.

16) 田中・高橋：" エンプティジョブの処理件数," 数理科学（1971年11月），50－54.

17) 栃内，他：" 処理件数による OSの能力測定について," 第3回研究セミナー報告，京大大型計算機センター（1971年12月），48－63.

18) 恒川・杉本：" FORTRAN 処理系のオプティミゼイションの評価." *

19) 亀田：" プログラムの評価に関する一考察," 第11回プログラミングポジウム報告集，（1970年1月）C1－10.

20) －：" CONTEST 方式を中心としたオペレーティングシステムの性能測定." *
19と同誌.

21) Joslin, E. O. & J. J. Aiken : " The validity of basing computer selections on benchmark results," Computers and Automation 15, 1 (Jan. 1966), 22－23.

22) Dopping, O. : " Test problems used for evaluation of computers," BIT 2, 4 (1962), 197－202.

23) Kenneth E. Knight : " Changes in Computer Performance," Datamation, Sept. 1966, pp. 40－54.

24) P. Calingert : " System Performance Evaluation ; Survey and Appraisal," CACM, Vol. 10, No. 1, pp. 12－18, Jan. 1967, Corrigenda on ibid., Vol. 10, No. 4, p. 224, April 1967.

25) Rubey, R. J., et al. : " PL/I comparative evaluation," American Data Processing, Inc. (1969).

26) Williams, Q. N., et al. : A methodology for computer selection studies," Computers and Automation 12, 5 (May 1963), 18－23.

27) Knuth. D. E. : " An empirical study of FORTRAN programs," SOFTWARE 1, 2 (1971), 105－133.

28) 亀田寿夫・恒川純吉：" テスト・プログラムによるシステム評価," 情報処理，Vol. 13, No. 11.

29) Raichelson, E., G. Collins, A Method for Comparing the Internal Operating Speeds of Computers, Communications of the ACM, Vol. 7, No. 5, May 1964, pp. 309－310.

30) Rubey, R. J., A Comparative Evaluation of PL/1, Datamation, Vol. 14, No. 12, December 1968, pp. 22－25.

31) Joslin, E. O., Application Benchmarks : The Key to Meaningful Computer Evaluations, Proceedings ACM 20th National Convention, 1965, pp. 27－37.

32) David 他：" Throughput measurement using a synthetic job stream," FJCC, 1971.

33) Arbuckle, R. A., Computer Analysis and Throughput Evaluation, Computers

and Automation, January 1966, pp. 12−15.

34) Buckley, F. J., Estimating the Timing of Workload on ADEP Systems: An Evaluation of Methods Used, Computers and Automation, February 1969, pp. 40−42.

Ⅳ Simulation, Program behavior

35) 北川, 金沢, 萩原："バッファ・メモリ・シミュレーション," 情報処理学会第12回大会, 1971.

36) 飯塚, "キャッシュ・メモリ・システム(I)," 情報処理, Vol. 13, No. 7, pp. 467−473, 1972.

37) 飯塚肇：キャッシュ・メモリ・システム, 情報処理, Vol. 13, No. 7, '72, pp. 467−473.

38) R. L. Maltson et al., : "Evaluation techniques for storage hierarchies," IBM Sys. J., Vol. 9, No. 2, pp. 78−117, 1970.

39) 淵一博ほか：ETSSの解析（その1）── データ収集・解析によるシステムの改良 ──, 電気試験所い報, Vol. 33, No. 12, pp. 53−74.

40) 淵, 田中, 真子, 弓場, 古川：ETSSの解析（その2）── GPSS／360による計算機操作システムの記述と解析について ── 電試い報, Vol. 34, No. 2, pp. 151−168, (1970).

41) 相磯：計算機システム・シミュレーションに関する研究. 電試研究報告, 第703号, (1969).

42) Kawai, H., Abrams, M. D. and Pyke, T. N., Jr. : Performance Measurement of Remote Access Computer Systems. 電総研い報, Vol. 34, No. 10, pp. 779−801, (1970).

43) 三上徹ほか：コンピュータ・システム性能評価シミュレータ PACSS, 情報処理Vol. 12, No. 1, pp. 14−25, 1971.

44) Noe, J. D. and Nutt, G. J. : Validation of a trace-driven CDC 6400 simulation. SJCC 1972, pp. 749−757.

45) Merikallio. R. A. and F. C. Holland. "Simulation Design of a Multiprocessing System," AFIPS Conference Proceedings. Vol. 33, part 11, (1968 FJCC). pp. 1399−1410.

46) Scherr, A. L. : "Time-Sharing Measurement." Datamation, Vol. 12, No. 4 (April, 1966), pp. 22−26.

47) Nielsen, Norman R. : "The simulation of timesharing systems." Comm. ACM 10, 7 (July 1967), 397−412.

48) ──. "Computer simulation of computer system performance." Proc. ACM 22nd Natl. Conf., 1967, MDI Publns., Wayne, Pa., pp. 581−590.

49) Cantrell, H. N. ; and A. C. Ellison. "Multiprogramming system performance and anal-

50) N R NIELSEN
The simulation of time sharing systems
COMMACM 107 July 1967

51) 金田；対話形・グラフィック・シミュレーション・システム. 情報処理, Vol. 13, No. 10, pp. 665−672, (1972).

52) Ihrer, F. C. ; A Technical description of the SCERT program. COMRESS, Inc., 5th Edition, (1967).

53) 益田, 高橋, 吉沢, "ページング・マシンにおけるスワッピング・アルゴリズムの比較とプログラムの動作解析," 情報処理, Vol. No. 13, pp. 81−88, 1972.

54) G. H. Fine et al., "Dynamic program behavior under paging," Proc. of the 21st ACM Nat. Conf., pp. 223−228, 1966.

55) P. J. Denning, "The working set model for program behavior," CACM, Vol. 11, No. 5, pp. 323−333, 1968.

56) 川合英俊："シミュレーションによるシステム評価法," 情報処理, Vol. 13, No. 11.

57) 益田隆司・高橋延匡："アドレス軌跡を利用した計算機システムの解析法," 情報処理, Vol. 13, No. 11.

V Monitoring

58) R. Sedgewick, R. Stone and J. W. Mcdonald : SPY−A program to monitor OS/360, Proc. 1970 FJCC, Vol. 37, pp. 119−128.

59) 富岡 他：汎用コンピュータ・パフォーマンス・アナライザ, 昭和46年度情報処理学会第12回大会予稿集, pp. 369−370.

60) W. R. Deniston : "SIPE: a TSS/360 software

61) 小野隆喜, 箱崎勝也：システム・データ収集装置− SYDAS, 昭和46年電子通信学会全国大会, No. 1053.

62) 箱崎勝也・小野隆喜：ハードウェア・モニタ (SYDAS) によるシステム性能評価, 情報処理学会第12回大会, No. 186, 1971.

63) D. T. Bordsen, "UNIVAC 1108 hardware instrumentation system," Proc. of the ACM workshop on system performance evaluation, pp. 1−28, 1971.

64) K. W. Kolence : A Software View of Measurement Tools, Datamation, Vol. 17, No. 1, pp. 32−38, 1971.

65) 箱崎勝也・小野隆喜："ハードウェア・モニタによるシステム測定," 情報処理, Vol. 13,

No. 11.

66) 橋本茂司：" ソフトウェア・モニタリングによるシステム測定," 情報処理, Vol. 13, No. 11.

67) 北川一 他：" オペレーティング・システムのパフォーマンス・モニタリング," 情報処理, Vol. 13, No. 11.

Ⅵ Analytic model

68) 田中穂積：並列循環待ち行列を用いたオンラインシステムの解析, 電子通信学会論文誌, Vol. 53−c, No. 10, pp. 756−764, (1970).

69) L. A. Belady, " A study of replacement algorithms for a virtual storage computer," IBM Sys. J., Vol. 5, No. 2, pp. 78−101, 1966.

3. 商用評価システム

3. 商用評価システム

本章では，第2章で眺めたコンピュータ・システムの評価技術のうち，現時点で，システム化あるいはパッケージ化され，商用システムとして利用者の便に供されている種々のシステムについて，その種類・機能・利用状況についての解説を試みたい。

ここにとりあげられている例は，もちろんこれらのシステムのすべてではなく，特に代表的と考えられるものであり，かつシステムに関する資料の入手が，比較的容易なものが中心になっている。

3.1 商用評価システムの概要

コンピュータ・システムのパフォーマンスを測定・評価する技術には，単にハードウェア性能のみを表すものから，システム全体のパフォーマンスを示すものまで，各レベルにわたって様々の手法，尺度があることは，前章で述べた。これらの技術のうち，商用システムとして世に出ているものを類別すると以下のように分けることができるであろう。

(1) アカウンティング・システムのレポータ

(2) モニタリング・システム

- ハードウェア・モニタ
- ソフトウェア・モニタ

(3) シミュレーション・システム

ここで，アカウンティング・システムのレポータというのは，コンピュータ・システムに備えられているアカウンティング・システムのロギング情報を入力として，適当なリダクションを施こしたあと，システムのユーティリゼージョン状態を計算し編集・出力するものである。第3世代以後のコンピュータでは，程度の差はあれ，かなり詳細な情報をアカウント・データとして各ジョブ・ステップごとにロギングしている。このような機能の代表的なものとしては，IBM 360/370 のSMF（Systems Management Facility）があげら

れる。

アカウント・データを利用することにより，システムのワーク・ロード特性やシステム全体のパフォーマンスの測定・評価が可能であるにもかかわらず，従来のアカウンティング・システムの不備，精度などの問題から多くの場合，あまり有効に利用されてはいない。

このようなアカウンティング・システムの情報をもとにしたコーティリゼーション・レポータとしてCIMSを例にとりあげ，第2節でその概要を紹介する。表3－1はアカウンティング・システムのレポータの一覧である。

表3－1　アカウンティング・システム・レポータ

名　　称	作　成（価格）＊	機　能　概　要
COMPUMETER	Computing Efficiency Inc. （$ 5,000）	OS 360／370のSMFを入力データとして各リソースのユーティライゼーションなどを分析しレポートを作成する。
CIMS	Boothe Computer Corp. （$ 5,300）	OS 360／370のSMFを入力データとし上記のCompumeterとほゞ同じような機能を持つ。

次に，コンピュータ・システムの評価技術のうち，かなり早くから適用化され，もっとも普及しているものとして，モニタリング・システムがあげられる。

モニタリング・システムは，稼動中のコンピュータ・システムに適当なプローブ（Probe）を接続し，あるリソースのユーティリゼーション状態に関する実測データを得るものである。この測定プローブが，ハードウェアでできているか，ソフトウェアでできているかによって，モニタリング・システムを，ハードウェア・モニタ，ソフトウェア・モニタの2種に分けることができる。これらのモニタリング・システムについては，第3節，第4節に詳説するが，どちらの型のモニタを利用するにしても，この方法はあくまでパフォーマンスに

関するデータの収集技術の一つであり，ここで得られたぼう大なデータをどのようにして処理し，解析するかが大きな問題となる。従ってこのようなシステムは，商用システムとして考えた場合，データ収集能力もさることながら，そのデータの解析用ソフトウェアのサポートがどの程度親切になされているかが，かなり大きな選択ポイントとなってくる。これらのモニタリング・システムを提供しているメーカでは，この解析用ソフトウェアの充実に大きな力を入れている。

表３－２はハードウェア・モニタ，表３－３はソフトウェア・モニタの一覧である。

（表３－３のソフトウェア・モニタには，システム全体を対象とするものと，利用者の，ある特定のプログラムのみをモニタリングするものとが混在しているが，詳細については第 3.4 にて述べる）

次に，評価技法のうち，もっとも強力で融通性に富んでいるものとしてシミュレーションがあげられる。

コンピュータ・システムのパフォーマンスを評価するのに用いられるシミュレーションにはイベント・オリエンティッドのものと，経験的に集めたファクタ・ライブラリをもとにして評価対象システムに対応させてシミュレートするものの２種類がある。また言語としてはFORTRANやPL／Ｉなどの汎用言語を用いてプログラムを作成する場合もあり，GPSSやSIMSCRIPT などのシミュレーション言語を用いてモデルを記述する場合もあるが，商用システムの多くはこれらの手続記述式（Procedural）のものではなく，むしろパラメータ指定方式の専用パッケージを利用する形式のものが多い。

表３－４は専用シミュレータの一覧である。

		Counters								
Model	Logic	Number/type	Width	Rate	Resolution	Probes	Peripherals	Prices	Maintenance	Software
CPMⅡ	450–600hub	16–32/hard	10 decimal	5–MHz	50–nsec	20–40	Tapes	$35,000	Allied Computer Technology	MSR(Summary report Program) CPS(Software monitor) (for both Monitors)
CPMⅢ	300–600hub	12–36/hard	10 decimal	10–MHz	25–nsec	16–48	Tapes, Plotter	$24,000		
CPMX	1200hub	20–48/hard	7 decimal	10–MHz	30–nsec	20–80	Tapes	$24,000		
ME–1011	148 hub	6/hard	6 decimal	10–MHz	30–nsec	8–16	Plotter Printer Tapes	$ 8,000	Boole & Babbage	MEDS(Data Summary Program)
X–RAY	480 hub	32/soft	32 bit	15–MHz	30–nsec	96	Tapes	$66,800	Teradata	Editor Analyzer
SUMⅠ	300 hub	16–32/hard	9 decimal	5–MHz	50–nsec	20–40	Tapes	$16,000	Computer Synectics Inc	SUMDAP (Data Analysis) Configuration simulator System Profiler
SUMⅡ				12.5–MHz	20–nsec			$21,500		
Dynaprobe										Dynapar (Data Analysis) Dynamap (for all Monitor)
D–7700	252 hub	16–18/hard	6 decimal	5–MHz	100–nsec	18	Printer	$10,512	COMRESS	
D–7800	320 hub	16/hard	10 decimal	10–MHz	30–nsec	32	Tapes	$25,000		
D–7900	600 hub	16/hard	10 decimal	10–MHz	30–nsec	32	Tapes	$27,000		
D–8011	600 hub	可変/soft	可 変	10–MHz	30–nsec	48	TTY Tapes	$23,000		
CMSM	240 hub	32/hard	9 decimal	10–MHz	30–nsec	48	Tapes Printer	$22,500	Copac, In	Measurement Software Package

表3－2　ハードウェアモニター一覧表

表3－3　ソフトウェア・モニタ一覧表

名　　称	作　成（価格）＊	機　　能　　概　　要
SMS	Boole & Babbage （$ 21,700）	IBM 360／370用，3つのシステム PPE ： 1本のプログラムを分析 CUE ： システム全体のバランスを測定し，スループットを向上させる。 DSO ： ディスク内のデータセットの配置を効率化する。 からなる。300社以上のユーザを持つ。
SUPERMON	S R I （$ 310）	OS 360用 サンプリングによりイベントデータを得る。測定のためのオーバヘッドを最少におさえているのが特長。レポート作成機能もある。
PROGLOOK	Stanford Linear Accelerator Center （$ 275）	OS 360 用でユーザプログラムの実行を測定し，サマリーレポートの図表化もできる。
LEAP	Lambda Corp	OS 360 用 実行時の時間分布を分析して種々のレポートを作成する。
M－ TEST	Computer Management Sciences	IBM 360 と Honeywell 200用 主としてCOBOLオブジェクトの効率に関するレポートを出力する。。
STAGE II	Tesdata	COBOLのソースプログラムを入力するとオプティマイズされたソースプログラムを出力する。
COBOL optimizer	CAPEX	COBOLのオブジェクト・プログラムを入力するとオプティマイズされたオブジェクトプログラムを出力する。IBM 360／370
FORMAX	Computer Synectics	FORTRAN プログラムの各ステートメントの使用頻度や実行時間を測定しレポートを作成する。

表３－４　専用シミュレータ一覧

名　　称	作　　成	機　　能　　概　　要
SCERT	COMRESS	work load を SCERT 言語で記述し，用意されている factor lfbrary と共にモデルをつくりあげ，主に確率的な積上げ計算によりシミュレートする。 （ 75,000 FORTRAN ステートメント ）
CASE	Sesdata	SCERT と同じように豊富な library を有し CASE 自身は FORTRAN で書かれている。解析モデルも含む。
CSS	IBM	GPSS とよく似た概念のコンピュータ専用シミュレータで，CSS 言語により細かくモデルの記述をおこないイベント・オリエンティドにシミュレートする。 （ IBM 360，370 用 ）
SAM	ADR	empirical なアプローチとユーザによるプログラミングを組み合わせたシミュレータで，かなり柔軟なモデルを作成できる。
AMAP	IBM	OS 360 の下でのトレース・データを利用し，ユーザ・ジョブおよびシステムの解析を行ない，さらにシステム構成をいろいろ変えながらシミュレートする。
CUSIM	Boole & Babbage	同社の PPE, CUE の結果を利用し，分析およびシミュレートをおこなう。

本節で，便宜上３種に分類した評価システムは，それぞれを単独で利用しても，それなりの成果を上げ得るのはもちろんだが，より効果を上げるには，これらのシステムを組み合わせて利用した方が良いのは言うまでもない。

3.2 アカウンティング・システムのレポータ

3.2.1 アカウンティング・データとその利用

第3世代以降のコンピュータ・システムには，マルチ・プログラミングの方式が本格的に取り入れられたこともあって，個々のジョブの料金計算のための情報を得ることを第一目的とするアカウンティング・ルーチンがOSに組み込まれるようになった。初期のアカウンティング・システムで得られる情報は，せいぜいCPU時間，ジョブ経過時間，印刷行数など，ごく限られたものであったが，最近のシステムではジョブ・ステップ毎，ある時間毎などに，かなりきめの細かいデータが得られるようになってきている。

現状のアカウンティング・システムから得られる情報は大別して次の3つに分けることができる。

(1) 識別データ（プログラム名，プロジェクト名等）

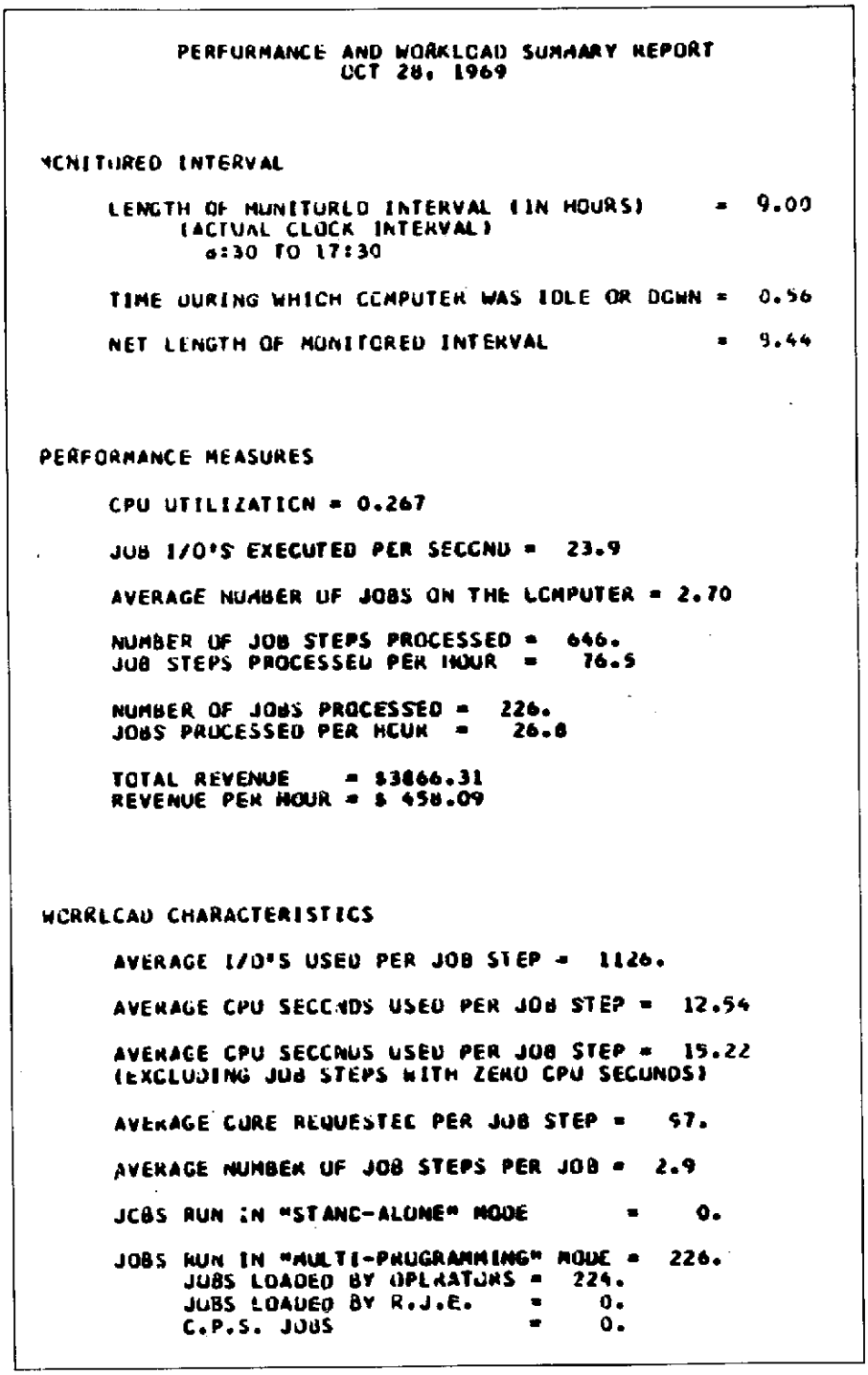
(2) 使用したリソース量（CPU時間，I／O回数，コア占有量など）

(3) 開始・終了時間

これらのアカウント・データは，利用のしかたによってコンピュータ・システムのパフォーマンス測定・評価に，非常に有用なデータとなり得る。特に既存システムの機器構成の変更の効果を評価するうえでは有効になる。

アカウント・データは，システムの各構成要素の利用率というより，各ジョブ毎に利用されたシステム・リソースの量を記録しているので，システム・パフォーマンスの評価よりむしろ，ワークロードの特性を評価することに向いているわけだが，システム全体のグロス・パフォーマンス・メジャーとして，パフォーマンス評価に利用することはできる。たとえば，コア利用率，CPU利用率，単位時間当りのI／O回数などのパフォーマンス・メジャーは比較的簡単に得ることができる。

アカウンティング・システムから得られる情報は，コンピュータの種類，OSとそのバージョンの種類，利用者によるアカウンティング・システムの変更などによって異なってくるが，一般には次のようなデータが記録されている。



```
          PERFORMANCE AND WORKLOAD SUMMARY REPORT
                    OCT 28, 1969


MONITORED INTERVAL

     LENGTH OF MONITORED INTERVAL (IN HOURS)      =  9.00
          (ACTUAL CLOCK INTERVAL)
            8:30 TO 17:30

     TIME DURING WHICH COMPUTER WAS IDLE OR DOWN =  0.56

     NET LENGTH OF MONITORED INTERVAL             =  9.44


PERFORMANCE MEASURES

     CPU UTILIZATION = 0.267

     JOB I/O'S EXECUTED PER SECOND =  23.9

     AVERAGE NUMBER OF JOBS ON THE COMPUTER = 2.70

     NUMBER OF JOB STEPS PROCESSED =  646.
     JOB STEPS PROCESSED PER HOUR  =   76.5

     NUMBER OF JOBS PROCESSED =  226.
     JOBS PROCESSED PER HOUR  =   26.8

     TOTAL REVENUE    = $3866.31
     REVENUE PER HOUR = $ 458.09


WORKLOAD CHARACTERISTICS

     AVERAGE I/O'S USED PER JOB STEP =  1126.

     AVERAGE CPU SECONDS USED PER JOB STEP =  12.54

     AVERAGE CPU SECONDS USED PER JOB STEP =  15.22
     (EXCLUDING JOB STEPS WITH ZERO CPU SECONDS)

     AVERAGE CORE REQUESTED PER JOB STEP =  97.

     AVERAGE NUMBER OF JOB STEPS PER JOB =  2.9

     JOBS RUN IN "STAND-ALONE" MODE       =    0.

     JOBS RUN IN "MULTI-PROGRAMMING" MODE =  226.
          JOBS LOADED BY OPERATORS =  224.
          JOBS LOADED BY R.J.E.    =    0.
          C.P.S. JOBS              =    0.
```

図３－１　Performance and Workload Summary Report

"Computer performance analysis : application of accounting data."より引用。

（以下のデータは各ジョブ・ステップ毎）

- 日付
- ジョブ番号
- 利用者番号
- ジョブ・ステップ番号
- コア要求量
- コア使用量および占有時間
- CPU時間
- 優先権
- プロセッサの種類（Compiler，linkなど）
- 開始時間
- 終了時刻

また比較的進んだアカウンティング・システムでは，次のようなデータも各ジョブ・ステップ毎に記録される。

- 完了コード（正常終了か異常終了かの区別）
- データ・セットの使用個数
- テープ・ディスクの数
- 入力カード枚数
- 各データ・セット毎のI／O回数

また，一日毎に次のようなサマリー・データが記録されているものもある。

- CPU wait 時間の総和
- IPL（システム・イニシェーション）の回数と時刻
- 処理したジョブ数
- 処理したジョブ・ステップ数
- ライン・プリンタに出力したデータ・セットの数

このようなアカウント・データを処理すれば，図3－1のようなパフォーマンスやワークロードのサマリー・レポートを得ることができる。

その他，コア占有量，CPU時間，I／O回数などに関するジョブ・ステップ数のヒストグラムも比較的簡単に得ることができる。

アカウント・データを利用し効果を上げ得る適用分野としては，以下のようなものが考えられる。

(1) システム変更の効果の測定・評価

(2) モニタリング・システムと組合せ，現在のシステムのパフォーマンスを測定し改善する。

(3) コンピュータ使用料金体系を定式化したり，改善したりする。

(4) パフォーマンス向上を計画する際のバック・グラウンド情報として利用する。

(5) コンピュータの利用率とワークロード特性の最新情報をインストレーション・マネージメントに役立てる。

(6) ワークロード特性の典型的なデータとして，シミュレーションやアナリティック・モデルの入力に利用できる。

(7) 過去における利用状況やパフォーマンスを理解できると同時に，将来の傾向の予測に役立つ。

このように，アカウント・データは，各種の目的に利用できるが，次に，前記(1)のシステム変更の効果測定を例にして，多数のベンチマーク・プログラムをコントロールしながら実行させる場合や，ハードウェア・モニタなどのモニタリング・システムを利用する場合と比較して，アカウント・データを利用する際の長短を整理してみると以下のようになる。

○利点

(1) ハードウェア・モニタやソフトウェア・モニタなどのように大金を投じて購入する必要がない。

(2) テスト・プログラムを特別な形で流すのではなく，通常のオペレーションでデータを得ることができる。

(3) 特別なテスト・プログラムのジョブ・ストリームをつくる必要はなく，日常のジョブ処理をそのまま利用できる。

(4) 処理されたジョブ群のワークロード特性が得られる。これは，ハードウェア・モニタやソフトウェア・モニタによっては得ることが比較的困難である。

(5) 第3世代コンピュータでは自動的にアカウント・データを収集してくれるので，収集費用が無料かあるいは安価である。

(6) アカウント・データは，通常自動的にシステムが記録してくれているのでシステム機器変更の効果を評価するのに，変更前の時点でのデータを解析するだけで利用できる。ハードウェア・モニタやソフトウェア・モニタの場合は，過去のある時点での変更に対する効果を測定しようと思うと，その時と同じようなハードウェア，ソフトウェアの状態，それにワークロード特性を再現させなければならない。

○欠点

(1) アカウント・データは，ハードウェア・モニタやソフトウェア・モニタから得られる測定データよりもかなりのチェック（たとえばエラー・ジョブなど）とリダクションが必要になる。

(2) ハードウェア・モニタやソフトウェア・モニタから得られる測定データに比べ，精度が劣る。

(3) アカウンティング・データでは，ハードウェア・モニタやソフトウェア・モニタなどのように，システムの各構成要素の利用状態（たとえば各チャネルの使用状況・各種キューの状況など）を測定することはできない。

(4) システム変更の効果に妥当な評価を下すには，かなり長期のアカウント・データをとり，解析しなければならない。（回帰分析などにより）

このようにアカウンティング・データのシステム・パフォーマンス評価に対する適用は十分とは言えないが使用上の手軽さの利点も含めてかなりの有用性に富んでいるが，この有用性が注目されだしたのは 1968 年頃からであり，その歴史は比較的新しい。

3.2.2 アカウンティング・システム・レポータの実例

A. CIMS*

CIMS（Computer Installation Management System）はBoothe Computer Corporationが提供しているアカウンティング・システムのレポータである。

CIMSは，IBM･OS 360/370でマルチ・プログラミングの効果と効率を正確に把握するためのデータ収集と解析をおこなうためのもので，インストレーションのマネージメントにかなり有用である。CIMSは主に米国内で，数十のカスタマに利用されている。

CIMSの入力データは，OS 360/370に備わっているアカウンティング・システムであるSMF（System Management Facility）の記録したファイルを利用し，各種のレポートを作成するようになっている。これらのレポートを作成する際，ある程度形式の定った標準形式のレポートを得るにはCIMSの出力機能を用いるが，利用者の要求に応じてバラエティに富んだレポート内容・形式を得るには，オプション機能のレポート・ライタを用いるようになっている。

CIMSの基本的な構成を図3－2に示す。

図3－2でSMF interface moduleはSMFで記録されたデータを処理しやすいようにフォーマット変換し，CIMS用のデータ・ベースとしてつくりあげるためのものである。

SMFからとり出す入力データとしては，各ジョブ・ステップ毎に以下のようなデータを得ている。

- ○日付
- ○開始時刻
- ○終了時刻
- ○要求優先権

＊CIMSについては，Boothe Computer Corporationより"System Description manul"他の提供を受け，図表などを引用させて頂いた。

○リジョン数

○コア占有量

○ジョブ名

○ジョブ・ステップ名

○プログラム名

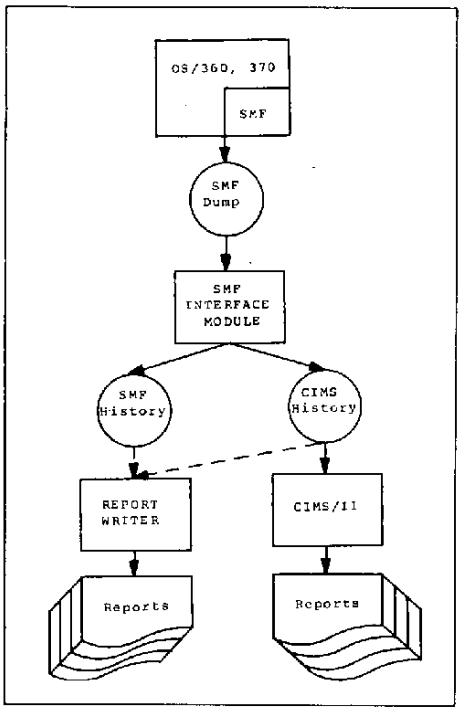


図３－２　CIMS 基本構成図

○完了コード

○システム出力量（SYSOUT）

○プログラム・ロード・アドレス

○CPU占有時間

○チャネル I／O 実行回数

○ドラム・ディスクの数とデータ・セット数

○磁気テープ台数とデータ・セット数

○各装置毎の入出力レコード数

○その他，インストレーション特有の追加情報

これらのデータをもとにして，ＣＩＭＳでは以下に示すように多種多様の管理用レポートを作成することができる。

(1) ジョブ会計情報

マルチ・プログラミングで実行されるジョブに対して常に正確な（公平な）料金を計算するのはかなり難しいが，ＣＩＭＳでは，ＣＰＵ時間，Ｉ/Ｏ時間量，コア，優先順位その他を考慮した標準的な料金計算アルゴリズムを設定し，各ジョブに対して適切な会計情報が得られるようになっている。

(2) プログラミング・アナリシス

プログラムの生産効率に関する参考情報の出力すなわち，テスト・ランの回数，コア占有量，ＣＰＵ時間，Ｉ／Ｏ時間，Ｉ／Ｏ依存度などのデータにより，プログラム・テストの手順の優劣，周辺機器の利用法，その他，プログラマーのプログラム作成技術・生産性などを判定し得るレポートを出力する。

(3) マルチ・プログラミングの効果

マルチ・プログラミングによりトータルのスループットをどれだけ上げているか，ＯＳの効率，ハードウェアの有効利用度などを評価するのに有用なレポートを作成する。

(4) オペレーション分析

マルチ・プログラミングによる効果は，オペレータの優劣により，かなり左右される。統計的解析により，オペレーション状態および各オペレータの成績などを正確に評価できるようなレポートを作成する。

(5) プログラム・プロファイルの分析

各プログラムのＣＰＵ時間，Ｉ／Ｏ時間等を統計的に分析し，Ｉ／Ｏバウンドかコンピュート・バウンドかなどのジョブ・クラスを判定し，マルチ・プログラミング効果を向上できるようなジョブ・ミックスの設定に役立つレポートを作成する。

(6) リソース利用状況分析

どのリソースが，どれだけ，どのジョブによって利用されているかなど

を分析する。

また，各ロード・モジュールの実行回数や，そのロード・モジュールがどんなリソースをどれだけ利用したかなどの詳細なレポートも作成する。

このような各種の機能を利用すれば，インストレーション・マネージメントに直接役立つ統計情報を得ることができる。

図３－３はリソース利用状況を示すレポートであり，図３－４はコア利用状況を示し，図３－５はそれをグラフで示したものであり，図３－６はあるインストレーションにおけるジョブの性格を示すプロファイルをグラフ化したものである。

BOCTHE RESOURCES INTERNATIONAL, INC.
COMPUTER INSTALLATION MANAGEMENT SYSTEM
RESOURCE MANAGEMENT REPORT

DATE	PROGRAM	START TIME	STOP TIME	..CORE... REQ	USED	CPU TIME	....I/C.... COUNT	TIME	ACT TIME	ELAPSED TIME	DISK UT	DS	TAPE UT	DS	UR	P R I	COMP CODE	RELATED ACCOUNTING DATA ***	Job accounting user fields
701201	IEBPTPCH	18628	18668	052K	052K	000.096	1148	0000.67	0000.77	0002.40	2	2	0	C	2	5	S000	CC111111	STEP1 000000000000000
701201	IEBPTPCH	20526	20569	052K	052K	000.108	1148	0000.67	0000.78	0002.58	2	2	0	0	2	5	S000	CC111111	STEP1 000000000000000
701201	IEBPTPCH	21715	21730	052K	052K	000.047	43C	0000.25	0000.30	0000.90	2	2	0	C	2	5	S000	CC111111	STEP1 000000000000000
701201	IEBPTPCH	22392	22398	052K	052K	000.018	24	0000.01	0000.03	0000.36	2	2	0	C	2	5	S000	CC111111	STEP1 000000000000000
701201	IEBPTPCH	22412	22417	052K	052K	000.023	24	0000.01	0000.03	0000.3C	2	2	0	0	2	5	S00C	CC111111	STEP1 000000000000000

↳ step name

***	RUNS	CPU TIME	I/O COUNT	I/O TIME	ACT TIME	ELA TIME	SYSIN	SYSOUT		
* TOTALS	5	000.292	2774	0001.61	0001.91	00006.54	0000000	00000030		*
*AVERAGE		000.058	555	0000.32	0000.38	00001.30	0000000	00000006	84%	I/O BOUND*
***										***

DATE	PROGRAM	START TIME	STOP TIME	..CORE... REQ	USED	CPU TIME	....I/C.... COUNT	TIME	ACT TIME	ELAPSED TIME	DISK UT	DS	TAPE UT	DS	UR	P R I	COMP CODE	RELATED ACCOUNTING DATA ***	Job accounting user fields
701201	IEQCBL00	23448	23469	130K	112K	000.498	1522	0000.89	0001.39	0001.26	2	6	0	C	1	3	S00C	CT050500	C08 000000000000000

***	RUNS	CPU TIME	I/O COUNT	I/O TIME	ACT TIME	ELA TIME	SYSIN	SYSOUT		
* TOTALS	1	000.498	1522	0000.89	0001.39	00001.26	0001167	00001217		*
*AVERAGE		000.498	1522	0000.89	0001.39	00001.26	0001167	00001217	64%	I/O BOUND*
***										***

DATE	PROGRAM	START TIME	STOP TIME	..CORE... REQ	USED	CPU TIME	....I/C.... COUNT	TIME	ACT TIME	ELAPSED TIME	DISK UT	DS	TAPE UT	DS	UR	P R I	COMP CODE	RELATED ACCOUNTING DATA ***	Job accounting user fields
701201	IEQCBL00	14148	14270	130K	112K	000.943	3711	0002.16	0003.10	0007.32	2	7	0	0	1	3	S004	CT053100	C08 000000000000000
701201	IEWL	14270	14287	140K	140K	000.057	263	0000.15	0000.21	0001.02	3	6	0	0	1	3	S000	CT053100	LKED 000000000000000
701201	PGM=*.DD	14287	14448	C52K	052K	000.161	109	0000.06	0000.22	0009.66	3	6	3	3	0	3	S001	CTC5310D	GO 000000000000000
701201	IEQCBLCO	23613	23683	130K	112K	000.908	3713	0002.17	0003.08	0004.20	3	7	0	C	1	3	S004	CT053100	C08 000000000000000
701201	IEWL	23683	23694	140K	140K	000.050	264	0000.15	0000.20	0000.66	3	6	0	C	1	3	S000	CT053100	LKED 000000000000000
701201	PGM=*.DD	23695	23921	C52K	052K	000.145	37	0000.02	0000.17	0013.56	3	6	3	3	0	3	S400	CTC5310D	GO 000000000000000

***	RUNS	CPU TIME	I/O COUNT	I/O TIME	ACT TIME	ELA TIME	SYSIN	SYSOUT		
* TOTALS	6	002.264	8097	0004.71	0006.98	00036.42	00C1949	00010389		*
*AVERAGE		000.377	1350	0000.79	0001.16	00006.07	0000324	00001731	67%	I/O BOUND*
***										***

DATE	PROGRAM	START TIME	STOP TIME	..CORE... REQ	USED	CPU TIME	....I/C.... COUNT	TIME	ACT TIME	ELAPSED TIME	DISK UT	DS	TAPE UT	DS	UR	P R I	COMP CODE	RELATED ACCOUNTING DATA ***	Job accounting user fields
701201	CT150101	21887	21890	C52K	C50K	000.002	3	0000.00	0000.00	0000.18	2	3	0	C	1	3	S000	CT150101	CT1501HI 000000000000000
701201	PAYLEDIT	21890	21914	120K	120K	000.023	107	0000.06	0000.08	0001.44	3	4	3	3	0	3	S000	CT150101	CT1501 00000000000000000
701201	CT150102	21914	21917	C52K	050K	000.002	3	0000.00	0000.00	0000.18	2	3	0	C	1	3	S000	CT150101	CT1501BY 000000000000000
701201	CT150101	22288	22292	052K	050K	000.002	3	0000.00	0000.00	0000.24	2	3	0	C	1	3	S000	CT150101	CT1501HI 000000000000000
701201	PAYLEDIT	22292	22408	120K	120K	001.621	10242	0005.97	0007.59	0006.96	3	4	3	3	0	3	SB37	CT150101	CT1501 00000000000000000
701201	CT150101	22476	22479	052K	050K	000.003	3	0000.00	0000.00	0000.18	2	3	0	C	1	5	S000	CT150101	CT1501HI 000000000000000
701201	PAYLEDIT	22479	22604	120K	120K	001.673	10243	0005.98	0007.65	0007.50	3	4	3	3	C	5	SB37	CT150101	CT1501 00000000000000000
701201	CT150101	23182	23186	C52K	050K	000.002	3	0000.00	0000.00	0000.24	2	3	0	C	1	3	S000	CT150101	CT1501HI 000000000000000
701201	PAYLEDIT	23186	23283	120K	120K	001.632	10245	0005.98	0007.61	0005.82	3	4	3	3	0	3	SB37	CT150101	CT1501 00000000000000000

***	RUNS	CPU TIME	I/O COUNT	I/O TIME	ACT TIME	ELA TIME	SYSIN	SYSOUT		
* TOTALS	9	004.96C	3C852	0017.99	0022.93	00022.74	0000003	00000003		*
*AVERAGE		000.551	3428	0002.00	0002.55	00002.52	0000000	00000000	78%	I/O BOUND*
***										***

NOTE DATA SELECTED ON DATE, RANGE 701130 TO 701201
SEQUENCED BY RELATED ACCOUNTING DATA,DATE,START TIME,
SUMMARY INFORMATION PRINTED ON BREAK IN RELATED ACCOUNTING DATA
I/O TIME = 35 MS X I/O COUNT
ACTIVITY TIME = I/O TIME + CPU TIME

*** SEQUENCED BY JOB NAME WITHIN RELATED ACCOUNTING DATA

図3－3

BOCTHE RESCURCES INTERNATIONAL, INC.
COMPUTER INSTALLATION MANAGEMENT SYSTEM
CORE UTILIZATION ACTIVITY LISTING

DATE	PROGRAM	START TIME	STOP TIME	..CORE... REQ	USED	CPU TIME	.....I/O..... COUNT	TIME	ACT TIME	ELAPSEC TIME	DISK UT	DS	TAPE UT	DS	UR	PRTY	COMP CODE	RELATED ACCOUNTING DATA	
701130	IFASMFDP	20131	2C194	052K	046K	000.214	4C80	0002.38	0002.59	0003.78	2	2	1	1	0	2	S000	ES280002STEP1	000000000000000
701130	IEHPROGM	20152	2C161	052K	052K	000.023	17	0000.01	0000.03	0000.54	2	4	0	0	0	5	S000	ES032005STEP	000000000000000
701130	PGM=*.DD	20191	2C423	100K	066K	010.954	4215	0002.46	0013.41	0013.92	3	7	1	1	1	5	S222	ES032005GO	000000000000000
701130	IEHPROGM	20265	2C323	052K	052K	000.138	83	0000.05	0000.18	0003.48	3	3	0	C	0	6	S000	FL703165SCRCH	CLEANLIB0000000
701130	MAPOSK	20323	2C348	052K	052K	000.414	626	0000.37	0000.78	0001.50	2	2	0	C	0	6	S000	FL703165SPACE	CLEANLIB0000000
701130	SC155800	20470	2C542	075K	050K	000.361	3304	0001.93	0002.29	0004.32	3	4	1	2	0	5	SBE8	SC159800STEP1	000000000000000
701130	IERRCOOO	20542	2C588	120K	110K	000.692	434C	0002.53	0003.22	0002.76	5	11	1	1	1	5	S000	SC159800STEP2	000000000000000
701130	SC155600	20613	2C684	065K	059K	001.464	3366	0001.96	0003.42	0004.26	3	3	6	6	0	5	S000	SC159600SC1596	000000000000000
701130	IERRCOOO	20656	2C814	120K	110K	000.819	4261	0002.49	0003.30	0007.08	5	11	1	1	1	5	S000	SC159900STEP1	000000000000000
701130	IERRCOOO	20710	2C747	120K	112K	000.072	193	0000.11	0000.18	0002.22	4	11	2	2	C	5	S000	SC151233SC1512	000000000000000
701130	IERRCOOO	20719	2C772	120K	112K	000.075	195	0000.11	0000.18	0003.18	4	11	2	2	0	5	S000	SC151234SC1512	000000000000000
701130	IEUASM	20797	2C829	096K	096K	000.454	517	0000.30	0000.75	0001.92	5	8	0	C	1	6	S008	FL703167ASM	DRVRQLRP0000000
701130	SC155900	20815	2C861	075K	052K	000.376	3248	0001.89	0002.26	0002.76	3	4	1	2	0	5	SFD0	SC159900STEP2	000000000000000
701130	SC155700	20862	2C921	100K	068K	000.333	757	0000.44	0000.77	0003.54	3	6	4	4	1	5	S000	SC155700SC1557	000000000000000
701130	SC153200	20976	2C996	065K	050K	000.040	253	0000.15	0000.19	0001.20	2	4	1	1	2	5	S000	SC153200SC1533	000000000000000
701130	IERRCOOO	21037	21C56	120K	112K	000.083	159	0000.09	0000.17	0001.14	4	12	1	1	1	5	S000	SC159700STEP97	000000000000000
701130	SC159700	21056	21C89	120K	C58K	000.042	71	0000.04	0000.08	0001.98	3	6	4	4	1	5	S000	SC159700SC1597	000000000000000
701130	IERRCOOO	21097	21126	120K	114K	000.092	163	0000.10	0000.19	0001.74	5	10	2	2	1	5	S000	SC223600SORT	000000000000000
701130	IEUASM	21112	21130	096K	096K	000.128	128	0000.07	0000.19	0001.08	4	8	0	0	1	6	S008	FL703168ASM	DRVRQLRP0000000
701130	IERRCOOO	21128	21156	120K	114K	000.082	162	0000.09	0000.17	0001.68	5	10	2	2	1	5	S000	SC223700SORT	000000000000000
701130	IEUASM	21131	21159	C96K	096K	000.47C	507	0000.30	0000.77	0001.68	4	8	0	C	1	6	S008	FL703169ASM	DRVRPPTR0000000
701130	IEUASM	21160	21170	096K	096K	000.128	128	0000.07	0000.19	0000.60	4	8	0	0	1	6	S008	FL703169ASM	DRVRPPTR0000000
701130	SC224500	21171	21201	065K	060K	000.050	51	0000.03	0000.08	0001.90	3	4	4	4	1	5	S000	SC224500SC2245	000000000000000
701130	IEBGENER	21216	21227	065K	066K	000.032	27	0000.02	0000.05	0000.54	2	3	1	1	1	5	S000	SC159799PTAMEX	0000LASH0000000
701130	IERRCOOO	21263	21311	120K	110K	000.742	426C	0002.49	0003.23	0002.88	5	11	1	1	1	6	S000	SC159900STEP1	DRVRPPTR0000000
701130	SC155900	21312	21347	075K	052K	000.364	3248	0001.89	0002.25	0002.10	3	4	1	2	0	6	SFD0	SC159900STEP2	DRVRPPTR0000000

図3－4

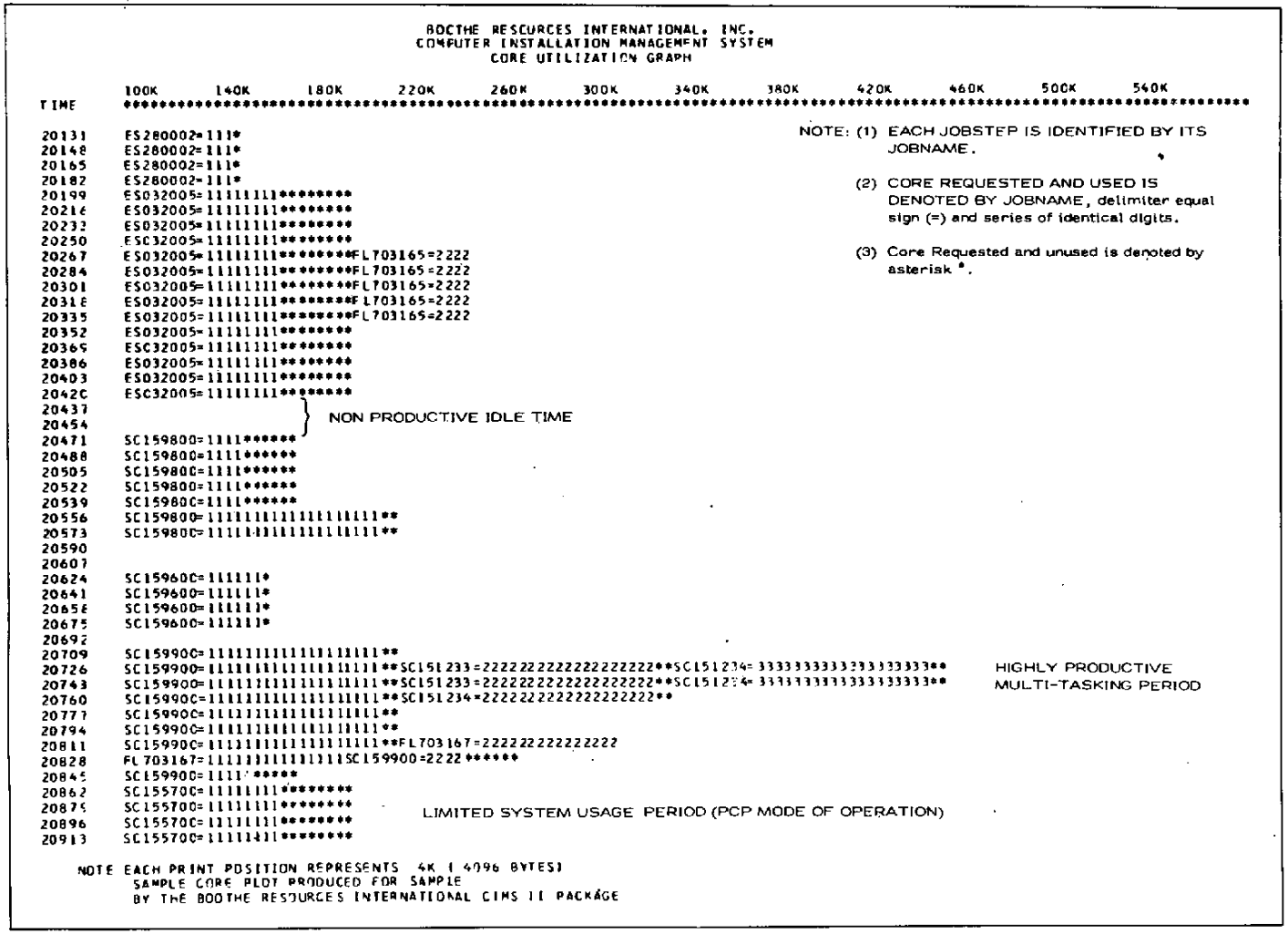
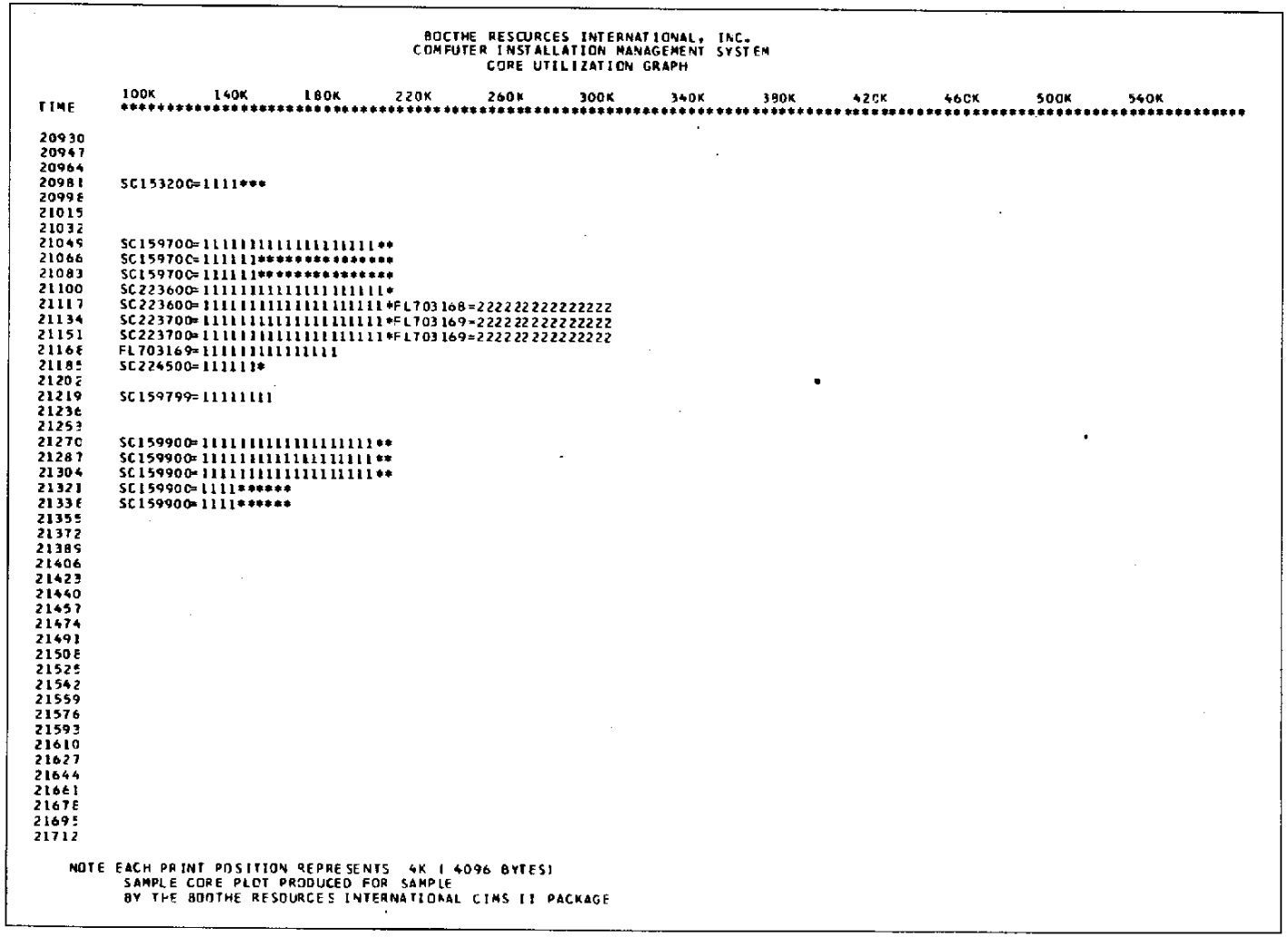

BOOTHE RESOURCES INTERNATIONAL, INC.
COMPUTER INSTALLATION MANAGEMENT SYSTEM
CORE UTILIZATION GRAPH
100K 140K 180K 220K 260K 300K 340K 380K 420K 460K 500K 540K
TIME
NOTE: (1) EACH JOBSTEP IS IDENTIFIED BY ITS JOBNAME.
(2) CORE REQUESTED AND USED IS DENOTED BY JOBNAME, delimiter equal sign (=) and series of identical digits.
(3) Core Requested and unused is denoted by asterisk *.
NON PRODUCTIVE IDLE TIME
HIGHLY PRODUCTIVE MULTI-TASKING PERIOD
LIMITED SYSTEM USAGE PERIOD (PCP MODE OF OPERATION)
NOTE EACH PRINT POSITION REPRESENTS 4K (4096 BYTES)
SAMPLE CORE PLOT PRODUCED FOR SAMPLE
BY THE BOOTHE RESOURCES INTERNATIONAL CIMS II PACKAGE

図 3－5



```
                                BOOTHE RESOURCES INTERNATIONAL, INC.
                              COMPUTER INSTALLATION MANAGEMENT SYSTEM
                                        CORE UTILIZATION GRAPH

          100K      140K      180K      220K      260K      300K      340K      380K      420K      460K      500K      540K
TIME      ********************************************************************************************************************

20930
20947
20964
20981     SC153200=1111***
20998
21015
21032
21049     SC159700=1111111111111111111**
21066     SC159700=11111***************
21083     SC159700=11111***************
21100     SC223600=1111111111111111111*
21117     SC223600=1111111111111111111*FL703168=222222222222222
21134     SC223700=1111111111111111111*FL703169=222222222222222
21151     SC223700=1111111111111111111*FL703169=222222222222222
21168     FL703169=11111111111111
21185     SC224500=11111*
21202
21219     SC159799=11111111
21236
21253
21270     SC159900=1111111111111111111**
21287     SC159900=1111111111111111111**
21304     SC159900=1111111111111111111**
21321     SC159900=1111******
21338     SC159900=1111******
21355
21372
21389
21406
21423
21440
21457
21474
21491
21508
21525
21542
21559
21576
21593
21610
21627
21644
21661
21678
21695
21712

    NOTE EACH PRINT POSITION REPRESENTS  4K ( 4096 BYTES)
         SAMPLE CORE PLOT PRODUCED FOR SAMPLE
         BY THE BOOTHE RESOURCES INTERNATIONAL CIMS II PACKAGE
```

図３－５の続き

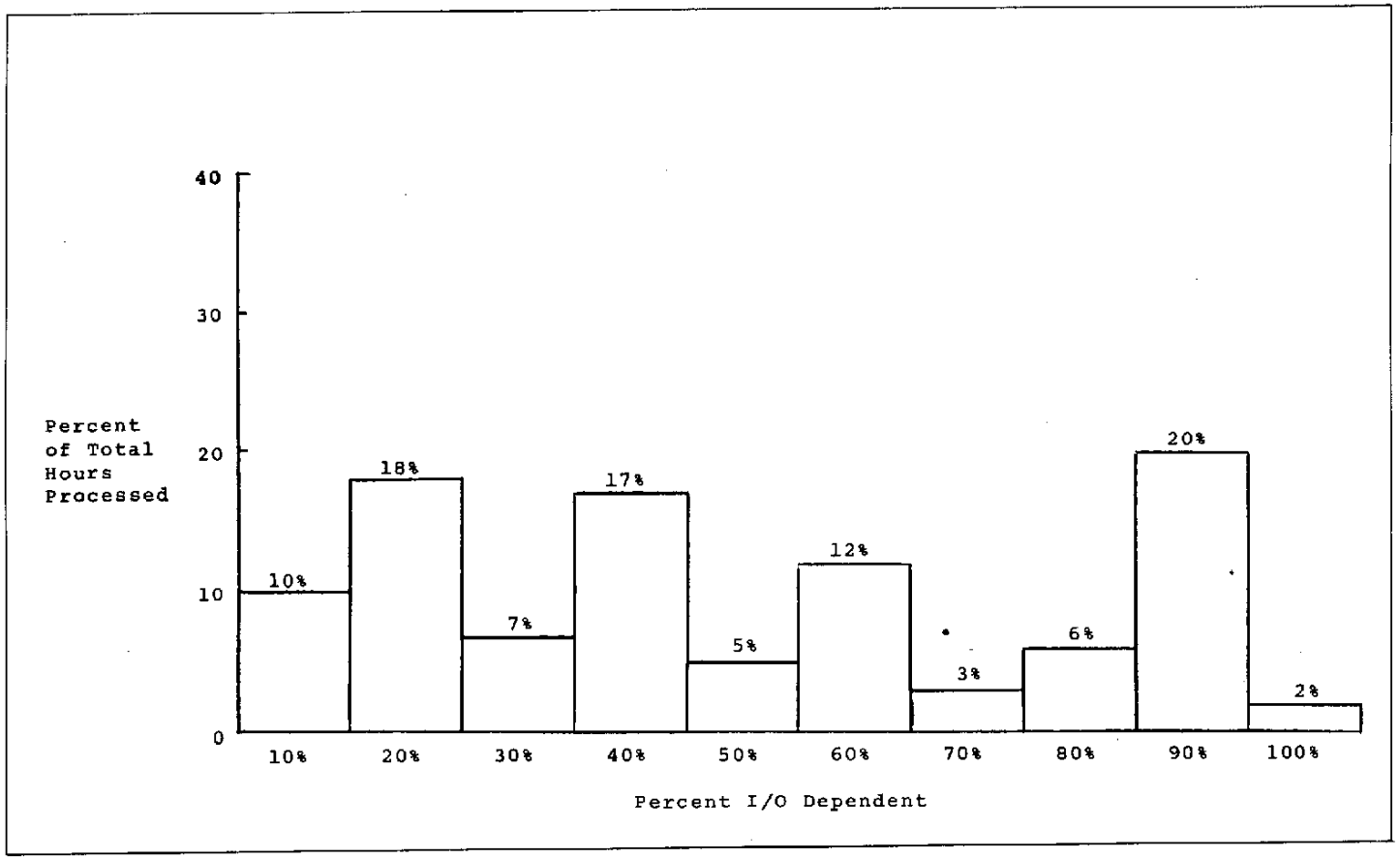


図3－6　COMPUTER INSTALLATION MANAGEMENT SYSTEM SHOP PROFILE

3.3 ハードウェア・モニタ

3.3.1 ハードウェア・モニタとその機能

A. ハードウェア・モニタとは

ハードウェア・モニタは，本質的には，コンピュータに関する時間・動作研究（生産管理上の用語）を行うための特別なストップ・ウォッチであると言うことができる。

実際に測定するためには，コンピュータの信号線部分に特別なセンサーを接続し，電気信号パルスの有無を検出するようになっている。たとえば，CPUがウェイト状態にある事を示す信号，すなわちCPUがアクティブ状態にはない事を示す信号などをセンサーで測定できるわけである。測定時には，コンピュータの信号に全く影響を与えないように考慮されており，干渉とか効率低下をおこすことなく，まるでオシロスコープで観測するがごとく測定できるようになっている。

検出した信号は，論理ロジックを持つパネル・ボードを経由させることにより，たとえば全 I／O 時間とか，I／OとCPUのオーバ・ラップなどのように組み合わせた機能として測定することもできる。普通，このパネル・ボードは自由にプログラムできるようになっており，適当な合成信号を得ることができる。

図３－７はその合成の様子を示したものである。

このようにして得られた信号は，カウンタに送られカウントされる。カウントしたデータは適当な表示機構で表示するか，磁気テープなどに記録しておいて，後にプログラムで適当な処理をおこない測定結果を印刷する。

図３－８にハードウェア・モニタを構成する要素と流れを示す。

このようなハードウェア・モニタは，ＩBMが早くから開発していたが，ＩBM以外の商用モニタが売出されたのは 1969年頃からである。

表３－５はハードウェア・モニタ開発の歴史の一部を示したものである。

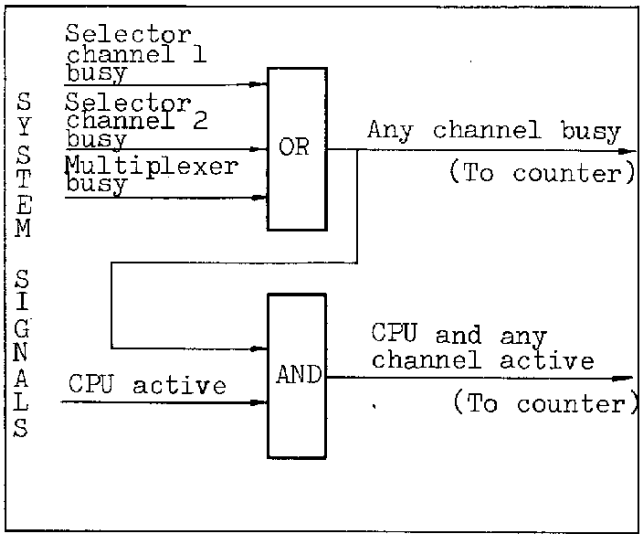


図３－７　合成信号

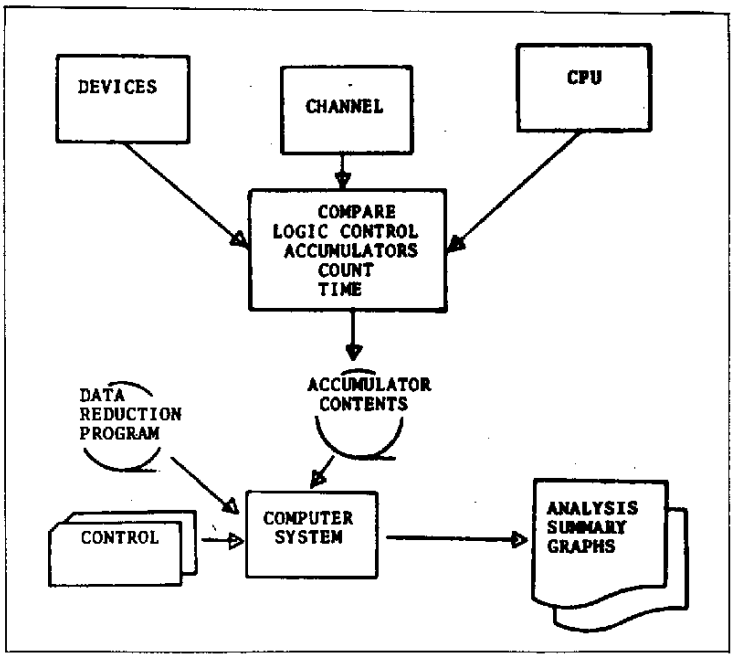


図３－８　ハードウェア・モニタ・システム

表3－5　ハードウェア・モニタの歴史（注1）

Year	Equipment	Source
1961	Channel Analyzer I/O channel & CPU wait time	IBM
1962	Program Monitor Full program trace on tape	IBM
1964	Program Event Counter (PEC) Channel & CPU overlap	IBM
1967	System Analysis Measuring Instrument (SAMI)—Measures all system resources, 4 tons (!), 64 counters, 32 comparators	IBM
1968	Basic Counting Unit (BCU) Transportable, limited system resource measurement, punched card output	IBM
1969	System Utilization Monitor (SUM) Portable, tape output, optional comparators, measures all system resources	Computer Synekties, Inc.

B. ハードウェア・モニタで測定できるデータ（注2）

ハードウェア・モニタを用いることにより，コンピュータ・システムの動作に関する種々のイベント・データが得られる。これらのイベントは，一個のセンサーで測定できるもの，複数個のセンサーで検出するもの，および，イベントの組合せとしての状態を測定するものの3種に分けることができる。以下にそれぞれの内容を示す。

(1)　1個のセンサーで測定できるイベント

以下の各イベントは1個のセンサーで測定できる。

CENTRAL PROCESSOR UNIT

① CPU STOP or MANUAL

（注1）　表3－5は文献2）より引用したものである。

（注2）　これらの測定項目は，文献4）より引用したものである。

② CPU WAIT

③ CPU RUN

④ MULTIPLEX CHANNEL BUSY

⑤ SELECTOR CHANNEL BUSY

⑥ PROGRAM CHECK INTERRUPT

⑦ I/O INTERRUPT

⑧ ALLOW INTERRUPT CHANNEL

⑨ PROBLEM STATE

⑩ SUPERVISOR STATE

⑪ INSTRUCTION FETCH

⑫ EXTERNAL INTERRUPT

⑬ CONSOLE BUSY

⑭ STATUS OF THE SYSTEM IN RELATION TO A PARTICULAR PROGRAM (IBM PSW PROTECTION KEYS)

DIRECT ACCESS STORAGE DEVICE

① CONTROL UNIT BUSY

② NUMBER OF SEEKS

③ INTERRUPT PENDING

④ READ BUSY

⑤ WRITE BUSY

⑥ DATA BUSY BY MODULE

⑦ TOTAL SEEK TIME BY MODULE

CONTROL UNITS

① DEVICE BUSY

② REWINDING TAPES

③ DATA TRANSFER

④ POLL of TERMINALS

UNIT RECORD EQUIPMENT

① LINES PRINTED

② CARDS READ

③ DEVICE BUSY

④ CARDS PUNCHED

(2) 複数個のセンサーを必要とするイベント

以下のようなイベントは複数個のセンサーで測定する。

① INSTRUCTION ADDRESSES

② REGISTER CONTENTS

③ INTERFACE DATA

④ PARTITION BOUNDARIES

⑤ DATA SET BOUNDARIES

(3) イベントの組合せ

イベントの組合せとして以下のようなものも測定できる。

① CPU ACTIVE

(CPU RUN∧CPU WAIT∧CPU MANUAL)

② ANY CHANNEL BUSY

(CHANNEL 1 BUSY∨CHANNEL 2 BUSY∨…∨CHANNEL N BUSY)

③ ANY CHANNEL BUSY ONLY

(ANY CHANNEL BUSY∧CPU WAIT)

④ TOTAL SYSTEM TIME

(CPU ACTIVE+SYSTEM WAIT)

⑤ CPU−CHANNEL OVERLAP

(CPU ACTIVE∧ANY CHANNEL BUSY)

⑥ CHANNEL OVERLAP

(CHANNEL 1 BUSY∧CHANNEL 2 BUSY…∧CHANNEL N BUSY)

⑦ SEEK ONLY
(CPU WAIT∧ANY CHANNEL BUSY∧SUM OF SEEKS IN PROGRESS ON ALL MODULES)

⑧ CPU ACTIVE ONLY
(CPU ACTIVE∧ANY CHANNEL BUSY)

⑨ CPU ACTIVE DURING PROGRAM STATUS WORD(CPU ACTIVE∧DECODE OF PSW)
e.g., PROBLEM STATE, SUPERVISOR, PROGRAM, etc.

⑩ I/O ACTIVE ONLY
(CPU RUN∧CPU WAIT∧CPU MANUAL)

⑪ SYSTEM WAIT
(CPU RUN∧CPU WAIT∧CPU MANUAL) or (TOTAL SYSTEM TIME-CPU ACTIVE)

⑫ SEEK WITHIN DATA SET
(SEEK∧WITHIN DATA SET BOUNDARIES)

⑬ INSTRUCTION WITHIN PARTITION
(INSTRUCTION FETCH WITHIN PARTITION BOUNDARIES)

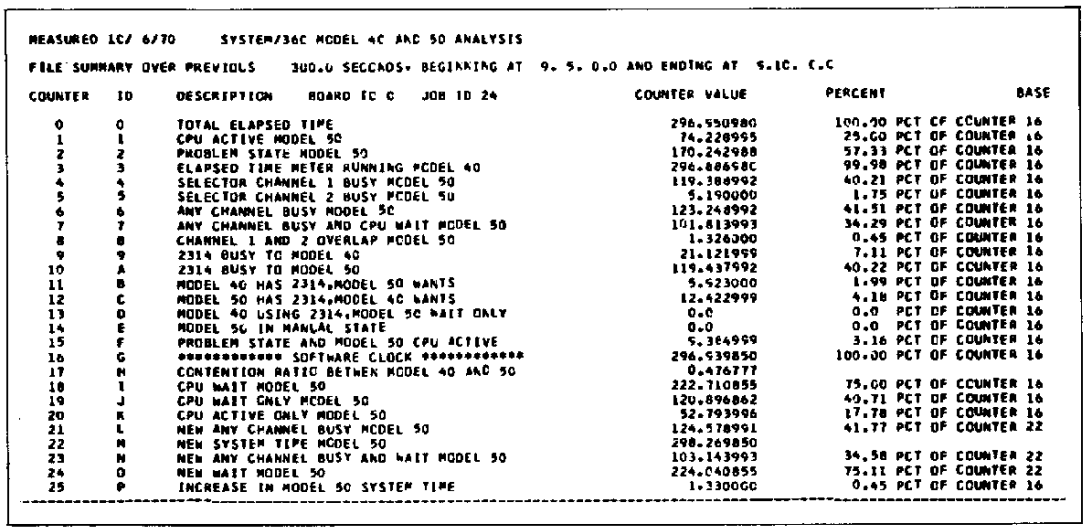
C. レポーティング用ソフトウェア

磁気テープなどに記録された測定データは，ハードウェア・モニタ・システムに用意されているレポーティング用ソフトウェアを用いて，評価用レポートとして出力することができる。

図3－9*はイベント・アクティビティの間隔のサマリ・レポート，図3－10はイベント・アクティビティの統計的なサマリ・レポート，図3-11はCPUがアクティブな状態のヒストグラム，図3-12はセレクタ・チャネル1がビジー状態のヒストグラム，図3-13はどれかのチャネルがビジー状態のヒストグラムを示すような例である。

---

*図3－9～図3－13は論文4）より引用したものである。



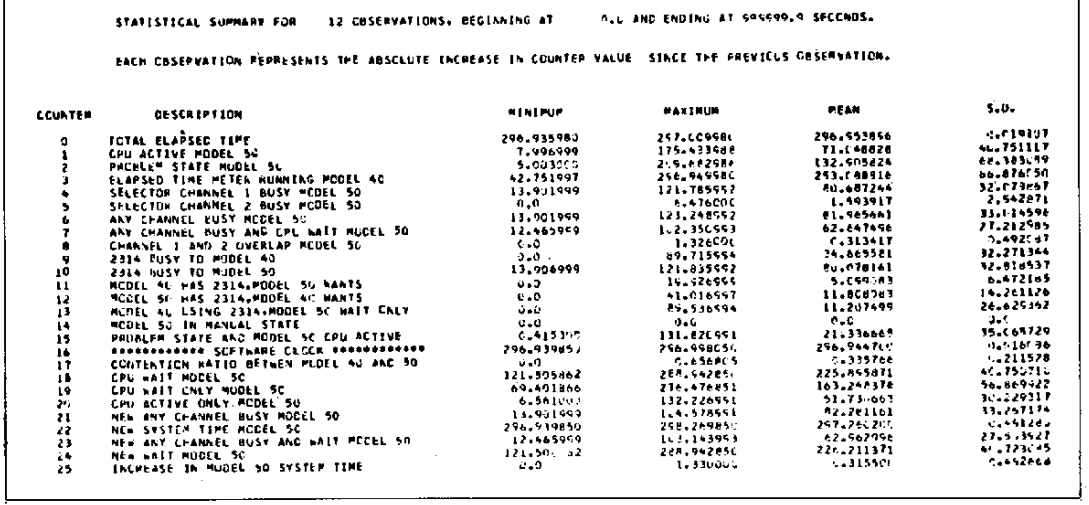
```
MEASURED IC/ 6/70      SYSTEM/360 MODEL 40 AND 50 ANALYSIS

FILE SUMMARY OVER PREVIOUS    300.0 SECONDS, BEGINNING AT  9. 5. 0.0 AND ENDING AT  9.10. 0.0

COUNTER   ID      DESCRIPTION     BOARD ID 0    JOB ID 24              COUNTER VALUE          PERCENT                BASE

   0       0      TOTAL ELAPSED TIME                                  296.550980      100.00 PCT OF COUNTER 16
   1       1      CPU ACTIVE MODEL 50                                  74.220995       25.00 PCT OF COUNTER 16
   2       2      PROBLEM STATE MODEL 50                              170.242988       57.33 PCT OF COUNTER 16
   3       3      ELAPSED TIME METER RUNNING MODEL 40                 296.886580       99.98 PCT OF COUNTER 16
   4       4      SELECTOR CHANNEL 1 BUSY MODEL 50                    119.388992       40.21 PCT OF COUNTER 16
   5       5      SELECTOR CHANNEL 2 BUSY MODEL 50                      5.190000        1.75 PCT OF COUNTER 16
   6       6      ANY CHANNEL BUSY MODEL 50                           123.248992       41.51 PCT OF COUNTER 16
   7       7      ANY CHANNEL BUSY AND CPU WAIT MODEL 50              101.813993       34.29 PCT OF COUNTER 16
   8       8      CHANNEL 1 AND 2 OVERLAP MODEL 50                      1.326000        0.45 PCT OF COUNTER 16
   9       9      2314 BUSY TO MODEL 40                                21.121999        7.11 PCT OF COUNTER 16
  10       A      2314 BUSY TO MODEL 50                               119.437992       40.22 PCT OF COUNTER 16
  11       B      MODEL 40 HAS 2314,MODEL 50 WANTS                      5.923000        1.99 PCT OF COUNTER 16
  12       C      MODEL 50 HAS 2314,MODEL 40 WANTS                     12.422999        4.18 PCT OF COUNTER 16
  13       D      MODEL 40 USING 2314,MODEL 50 WAIT ONLY                0.0              0.0  PCT OF COUNTER 16
  14       E      MODEL 50 IN MANUAL STATE                              0.0              0.0  PCT OF COUNTER 16
  15       F      PROBLEM STATE AND MODEL 50 CPU ACTIVE                 9.364999        3.16 PCT OF COUNTER 16
  16       G      ********** SOFTWARE CLOCK **********                296.539850      100.00 PCT OF COUNTER 16
  17       H      CONTENTION RATIO BETWEN MODEL 40 AND 50               0.476777
  18       I      CPU WAIT MODEL 50                                    222.710855       75.00 PCT OF COUNTER 16
  19       J      CPU WAIT ONLY MODEL 50                               120.896862       40.71 PCT OF COUNTER 16
  20       K      CPU ACTIVE ONLY MODEL 50                              52.793996       17.78 PCT OF COUNTER 16
  21       L      NEW ANY CHANNEL BUSY MODEL 50                        124.578991       41.77 PCT OF COUNTER 22
  22       M      NEW SYSTEM TIME MODEL 50                             298.269850
  23       N      NEW ANY CHANNEL BUSY AND WAIT MODEL 50               103.143993       34.58 PCT OF COUNTER 22
  24       O      NEW WAIT MODEL 50                                    224.040855       75.11 PCT OF COUNTER 22
  25       P      INCREASE IN MODEL 50 SYSTEM TIME                      1.330000        0.45 PCT OF COUNTER 16
```

図３－９　イベント・アクティビティ間隔



```
          STATISTICAL SUMMARY FOR    12 OBSERVATIONS, BEGINNING AT       0.0 AND ENDING AT 596699.9 SECONDS.

          EACH OBSERVATION REPRESENTS THE ABSOLUTE INCREASE IN COUNTER VALUE  SINCE THE PREVIOUS OBSERVATION.

COUNTER       DESCRIPTION                                  MINIMUM              MAXIMUM              MEAN              S.D.

   0      TOTAL ELAPSED TIME                          296.935980           297.CC9980           296.552856          0.0[illegible]
   1      CPU ACTIVE MODEL 50                           7.996999           175.433688           71.[illegible]      44.751117
   2      PROBLEM STATE MODEL 50                        5.003000           295.[illegible]      132.[illegible]     88.[illegible]
   3      ELAPSED TIME METER RUNNING MODEL 40          42.751997           296.[illegible]      253.[illegible]     66.[illegible]
   4      SELECTOR CHANNEL 1 BUSY MODEL 50             13.901999           121.[illegible]      80.[illegible]      32.[illegible]
   5      SELECTOR CHANNEL 2 BUSY MODEL 50              0.0                 8.[illegible]        1.[illegible]       2.542871
   6      ANY CHANNEL BUSY MODEL 50                    13.901999           121.248552           81.[illegible]      33.[illegible]
   7      ANY CHANNEL BUSY AND CPU WAIT MODEL 50       12.465999           102.[illegible]      62.[illegible]      27.[illegible]
   8      CHANNEL 1 AND 2 OVERLAP MODEL 50              0.0                 1.326000             0.[illegible]       0.[illegible]
   9      2314 BUSY TO MODEL 40                         3.0                 89.[illegible]       24.[illegible]      22.271344
  10      2314 BUSY TO MODEL 50                        13.906999           121.835992           80.[illegible]      32.[illegible]
  11      MODEL 40 HAS 2314,MODEL 50 WANTS              0.0                 16.[illegible]        5.[illegible]       6.[illegible]
  12      MODEL 50 HAS 2314,MODEL 40 WANTS              0.0                 41.016997           11.[illegible]      14.261[illegible]
  13      MODEL 40 USING 2314,MODEL 50 WAIT ONLY        0.0                 89.[illegible]       11.207499          26.625[illegible]
  14      MODEL 50 IN MANUAL STATE                      0.0                 0.0                  0.0                 0.0
  15      PROBLEM STATE AND MODEL 50 CPU ACTIVE         0.[illegible]       131.[illegible]      21.[illegible]      35.[illegible]
  16      ********** SOFTWARE CLOCK **********       296.[illegible]      296.[illegible]      296.[illegible]     0.[illegible]
  17      CONTENTION RATIO BETWEN MODEL 40 AND 50       0.0                 0.[illegible]        0.[illegible]       0.211578
  18      CPU WAIT MODEL 50                           121.[illegible]       288.[illegible]      225.[illegible]     44.[illegible]
  19      CPU WAIT ONLY MODEL 50                       69.401866           276.[illegible]      163.[illegible]     56.[illegible]
  20      CPU ACTIVE ONLY MODEL 50                      6.[illegible]       132.[illegible]      51.[illegible]      30.[illegible]
  21      NEW ANY CHANNEL BUSY MODEL 50                14.901999           14.[illegible]       12.[illegible]      1.[illegible]
  22      NEW SYSTEM TIME MODEL 50                    296.[illegible]      298.[illegible]      297.[illegible]     0.[illegible]
  23      NEW ANY CHANNEL BUSY AND WAIT MODEL 50       12.465999           103.[illegible]      62.[illegible]      27.[illegible]
  24      NEW WAIT MODEL 50                           121.[illegible]      288.[illegible]      226.[illegible]     41.[illegible]
  25      INCREASE IN MODEL 50 SYSTEM TIME              0.0                 1.330000             0.[illegible]       0.[illegible]
```

図３－10　イベント・アクティビティの統計的サマリ

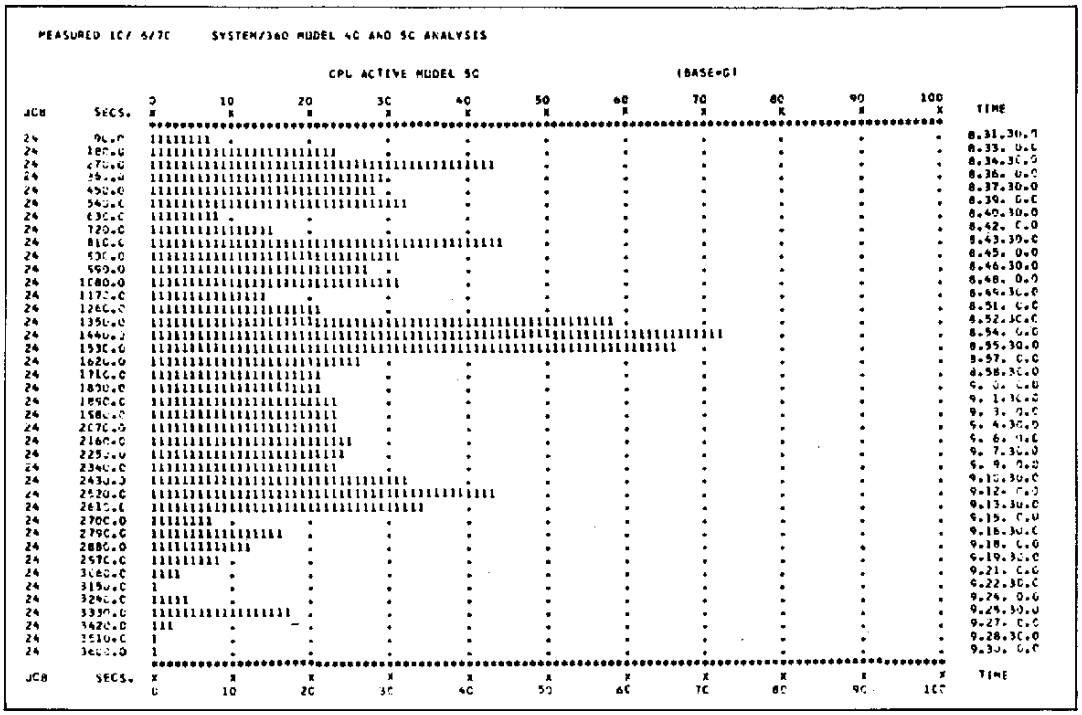


図3－11 －Histogram of CPU active

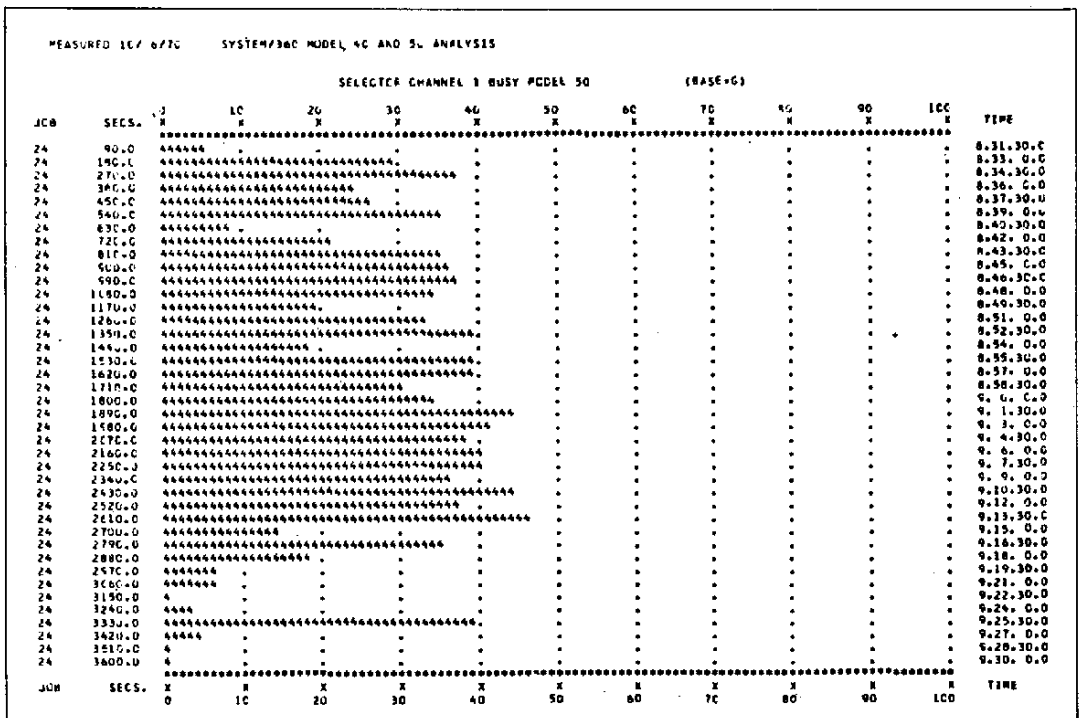


図3－12 －Histogram of selector channel 1 busy

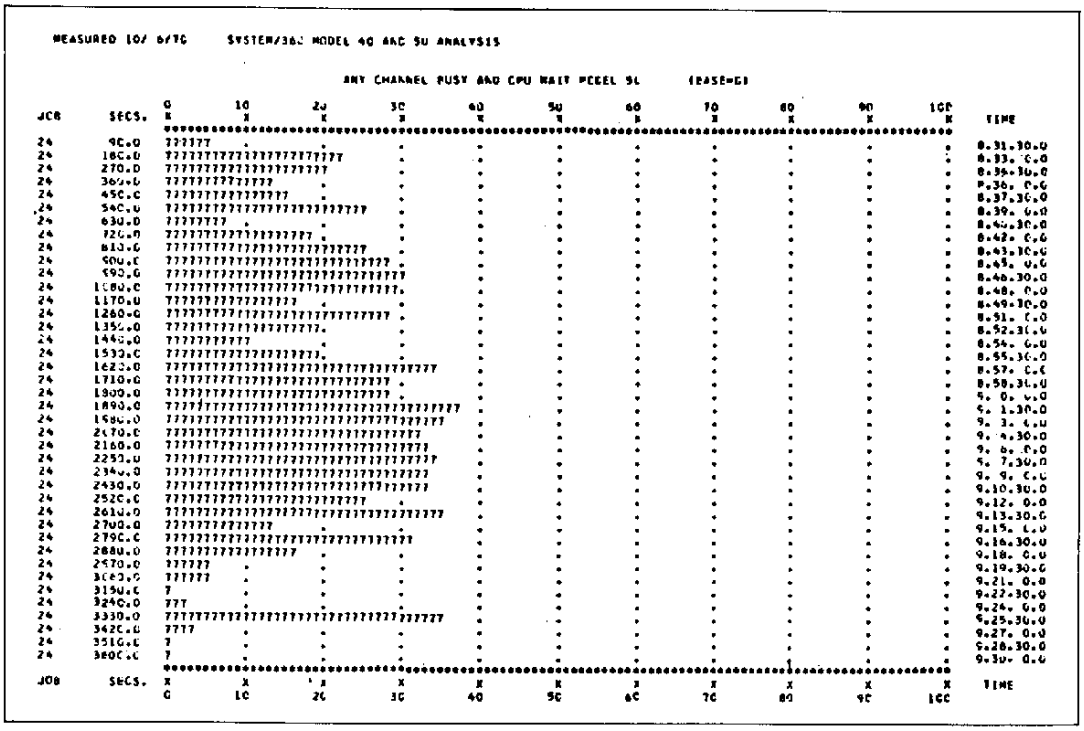


図 3－13 －Histogram of any channel busy only

D. ハードウェア・モニタの選択基準

ハードウェア・モニタには表3－2（P84）に示したように，各種のシステムがあるが，これらのハードウェア・モニタの選択にあたり考慮しなければならないポイントとしては以下のようなものがある。

(1) リゾルーションとリピテーション・レート

(2) モニタ・クロック

(3) カウンタの数，桁数など

(4) プローブの数・性能

(5) コンパレータの有無

(6) ロジック・パネル

(7) 出力方法

(8) 解析用ソフトウェア

(9) 技術的なサポート

(10) 価格

上記の項目中で主なものについてUNIVACのHart氏が論文 3） で解説を加えているので以下にそれを引用する。

○ リゾルーションとリピテーション・レート

ハードウェア・モニタのリゾルーションとは，カウンタが検出可能な電気信号の最小パルス巾のことで，リピテーション・レートとは1秒間にカウントできる最大数をいう。測定対象のシステムの信号速度・巾がモニタの能力を超えてしまう場合，測定結果は意味のないものになってしまうので注意を要する。したがって測定対象のシステムの信号速度によって，それにあったモニタを選択する必要がある。一般に，高速の回路は高価なので，高速のハードウェア・モニタになる程高価になってくる。

○ モニタ・クロック

クロックの役割は，モニタリングの自動的な動作の為のパルスを発生させることと，タイム・モードで働くカウンタに対するパルスの供給の2つであろう。クロックは，通常，水晶発振器でドライブされ，ある決ったレ

ートで動作するようになっている。また，モニタリングの間隔を事前にセットできるようなマスタ・クロック・カウンタがあり，設定された時間間隔が過ぎるとモニタリング回路がストップするようになっている。このクロック時間は，ダイヤルなどで 10 進値をセットできるようになっているが，中には，二進値のスィッチが並んでいるのもあり，この場合は，セットも難しいし，まちがいやすい。クロック・レートとしては，少なくとも 10 MHz のオーダは欲しい。たとえば，0.1 μsec のリゾルーションを得ようとすれば 10 MHz のクロックが必要である。また精度としては少なくとも 0.01％の振れにとどめていないと，十分正確な時間測定はできない。24 時間クロックが装備されているモニタもあり，24 時間，オペレータの介入なしで測定できる。いずれにしろ，高速のクロック・レートのモニタは測定間隔に見合うだけの十分な桁数のクロック・カウンタを必要とし，遅いクロックでは，カウンタの桁数は少なくてすむが，リゾルーションは悪くなり，このあたりのかねあいが問題である。

○　カウンタの数・桁数など

必要とされるカウンタの数は，もちろん測定対象のコンピュータ・システムによって違ってくる。

ハードウェア・モニタは通常モジュール構造になっているので，基本構成にオプションとして追加することにより，カウンタの数は，ある程度まで増やすことができる。

カウンタの数としては，8～10 個ぐらいあれば，小システムの評価には十分役立つ測定データを得ることができる。システム全体の測定というのは，通常適当な部分に分割することが可能であるから，現在使えるカウンタをうまく利用することである。カウンタの桁数としてはオーバフローの頻繁さをさけるため少なくとも 10 進 6 桁は欲しい。

○　プローブについて

測定対象のコンピュータ回路に直接接続されるプローブは，そのコンピュータとコンパティブルでなければならない。ハード・モニタの多くは正

しく接続できるように考慮してあり，ホスト・システムに悪影響を与えないように，なんらかのプロテクション機能を有している。通常は，ハイ・インピーダンスの差動アンプでできており，コンピュータに干渉がでないようにしてある。

また，測定電圧範囲も適当に調整可能になっており，大低のコンピュータは測定可能になっている。

ケーブルは，検出した信号を外乱なしに送ることを考慮しなければならないため，長さに制限がある。コンピュータ室の構成によっては，かなり長いケーブルを必要とするため，十分な長さと，雑音および信号の減衰がないケーブルを選ぶ必要がある。プローブの数は，各メーカの基本構成ごとに異なるが，追加プローブは，$ 50 ぐらいで手に入れることができる。

○ ロジック・パネルについて

効果的な測定データーを得るためには，ロジック・パネルは必須のものである。AND，NAND，OR，XOR，NOR，INVERTなどのロジックがあるが，これらの数は，通常オプションで増やせるようになっている。

複雑なマルチ・プログラミング・システムやTSSのシステム測定を効果的におこなうためには，通常，かなりのロジックが必要になってくる。

3.3.2 ハードウェア・モニタの実例

ハードウェア・モニタの実例としてTesdata社のX−RAYとComputer Synektics社のMicro SUM を示す。

X−RAYは，豊富な機能を持つかなり高価なシステムの一つであり，Micro SUMは現在商用になっているハードウェア・モニタの中では，多分，最もコンパクトで安価なシステムである。

A. *X−RAY

(1) X−RAYの概要

X−RAY（Execution−Recorder Analyzer）は米国のTesdata社が開発したハードウェア・モニタである。非常に高価ではあるが（$66,800），商用ハードウェア・モニタの中では最高級のレベルに位置するものの一つであると言える。

X−RAYは，コンピュータ・システムの各ハードウェア装置の利用状況を測定するだけではなく，OSのオーバヘッドに関する統計や，各アクティビティの活動状況に関する測定，アプリケーション・プログラムの構造とか，命令コードの効率に関する測定，それにファイル構造やアクセス・オーバ・ヘッドなどデータ・ハンドリングに関する測定も可能で，これらを解析して各種のレポートを作成するようになっている。

このような種々の機能により，コンピュータ・システム運用上の非効率的な部分を指摘することができるので，たとえばコンフィギュレーションのバランシング，OSの最適構成の決定，問題向けプログラムの構造再編，データ・ベースの再構成，プログラムの命令の最適化などの要求に即座に利用できるレポートを得ることができるとTesdata社では言っている。

X−RAYは，ハードウェア・モニタ本体であるX−RAY Recorderおよび，測定したデータを解析するソフトウェアから成っている。

---

*X−RAYについてはTesdata社より"X−RAY general information manual"の提供を受け，図表などを引用させて頂いた。

(2) X-RAY Recorderの構成

X-RAY Recorder の全体図および論理構成図を図3-14，図3-15に示す。

図3-15.にも示されるようにX-RAY Recoraerは以下のような部分から構成されている。

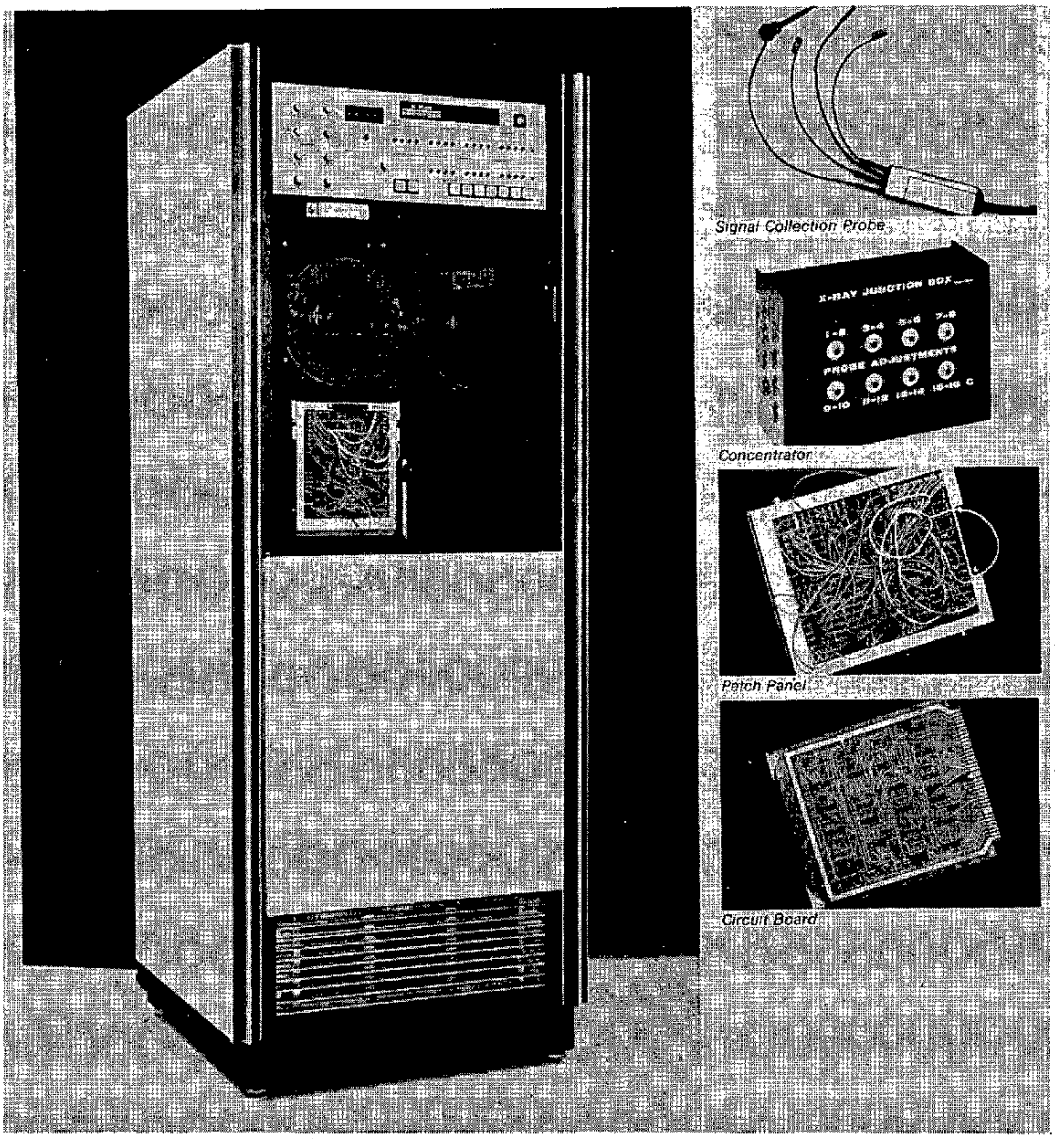


図3－14　X－RAY－Recorder

① 信号をとり出すプローブ（センサー）とコンセントレータ

接続できるプローブの数は 96 個までで，測定のリゾルーション は 30 ns 以上，リピテーション・レートは 15 MHz までである。

プローブは 16 ケーブルづつ コンセントレータでまとめられて Recorder のパッチ・パネル部分に接続されており，パッチ・パネルからプローブまでのケーブル最大長は 225 フィートである。

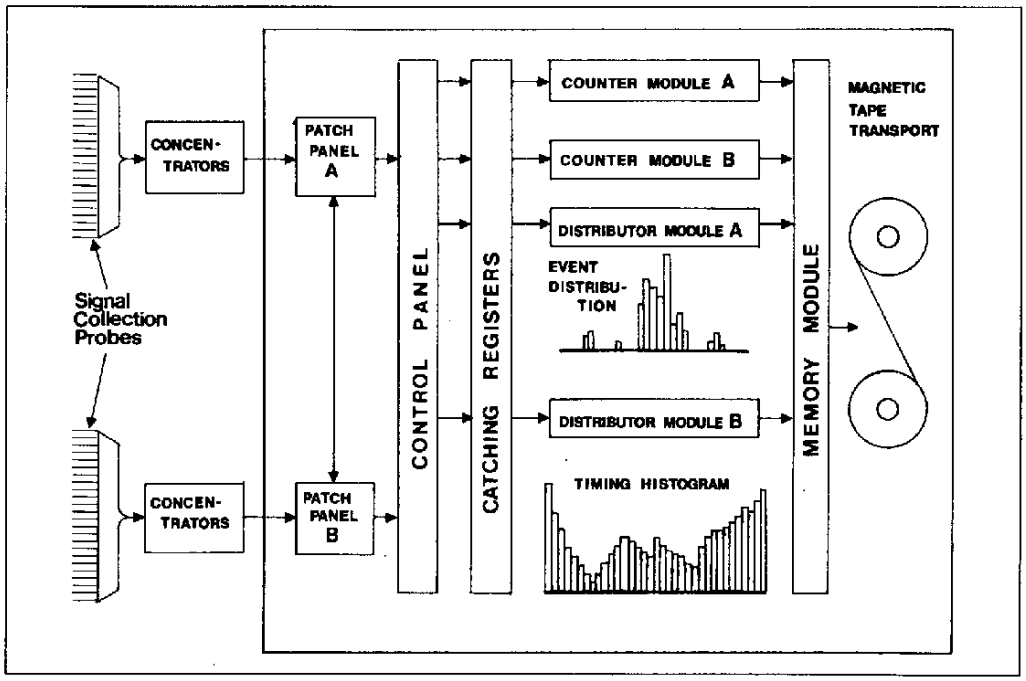


図 3－15　X-RAY　Recorder 機能図

測定できる電圧の範囲は±0.5から±5ボルトで，測定対象のコンピュータの種類によって調整できるようになっている。

② パッチ・パネル

コンセントレータからの信号は，パッチ・パネルで指定されたロジックにより組み合わされ，カウンタ・モジュールへの入力となる。このパッチ・パネルには，480のハブがついており，ワイヤの配線を変えることにより，かなり自由なロジックを組む事ができる。このロジック要素の構成は以下のようになっている。

Logical Element	Number of Patch Panels	
	One	Two
Hexadecimal Decoders	2	4
Inverters	12	24
Fanouts	14	28
Latches	4	8
Single Shots	2	4
AND Elements	20	40
OR Elements	12	24
Divider Networks	8	16
TRUE Elements	10	20

③ カウンタ・モジュール

カウンタ・モジュールは，信号のデュレーションやパルスの数をカウントするもので，パッチ・パネルからの入力信号により，測定対象でハードウェアの各装置個々にあるいは，ロジカルに関連させて（CPU，チャネル，コントローラ，各デバイス他）そのアクティビティを調べたり，ある特定のイベント（disc seek，tape write，terminal

active など）の発生を調べるのに用いるものである。

Recorder には 2 セットまでのカウンタ・モジュールを持たせることができ，1つのカウンタ・モジュールにつき 16 個の独立したカウント用のレジスタがつくようになっている。

このレジスタはメモリ・モジュール内の 32 bit のメモリで実現されており，このメモリの内容を，アリスメティック・モジュールによって更新することも可能である。

図 3−16 は，カウンタ・モジュールの機能を示したものである。

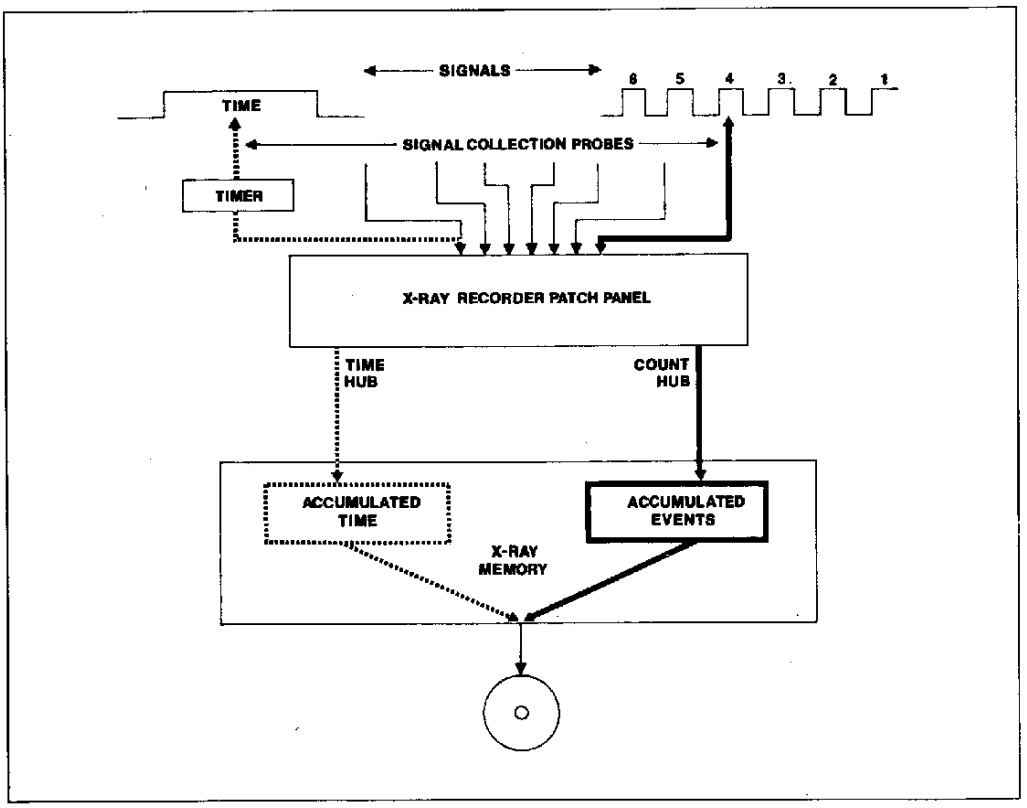


図 3− 16　カウンタ・モジュール機能図

④　ディストリビュータ・モジュール

ディストリビュータ・モジュールは，パッチ・パネルからの信号を受けて，データ入力の量や信号時間の度数分布をメモリ・モジュール内に

つくりあげるものである。特定の処理やアクセスに関する主記憶やディスクのアドレスを直接，メモリ・モジュール内に記録したりすることにより，あるプログラムの命令実行のマッピングやデータ・ベース内のあるエレメントのアクティビティのマッピングなどもできるようになっている。

Recorder には 2 セットのディストリビュータ・モジュールを持つことができ，各モジュール毎に，16 までの信号をパラレルに受けることができる。

また，このディストリビュータ・モジュールは，以下のような 3 種のモードで動作できるようになっている。

a) Store mode

このモードでは，ディストリビュータ・モジュールの入力をそのままメモリ・モジュールに転送する。磁気テープに記録している間も連続的な転送を許すため，896 ワードのバッファを用意している。サンプリングのレートは，ある固定した時間か，あるいは外部からの信号によって制御できるようになっている。

図 3-17 は Store mode の動作状態を示している。

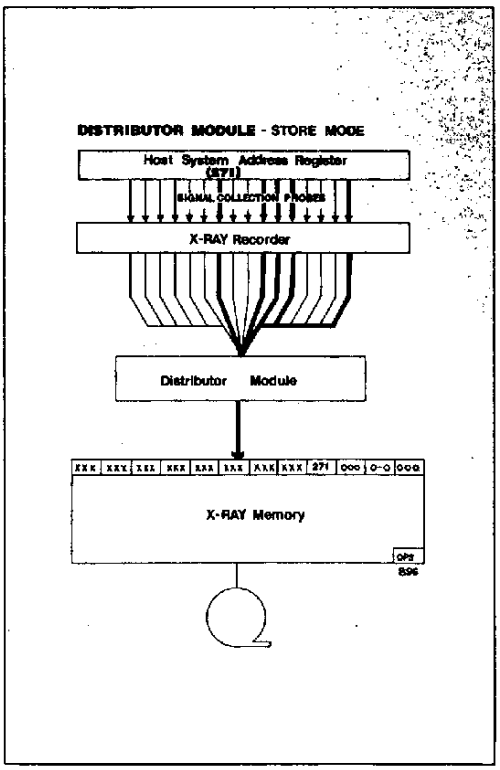


図 3－17　Store mode の動作状態

b） E−Map mode

このモードは，パッチ・パネルからの信号の大きさに従って，メモリ・モジュール内にその信号の度数分布をつくりあげるものである。

入力信号は適当なレートでサンプリング可能で，許容範囲もあらかじめセットすればチェック可能であり，512パートのリゾルーションでスケーリングできるようになっている。連続的なイベント分布を測定可能とするため512ワードの交替バッファ領域を用意している。

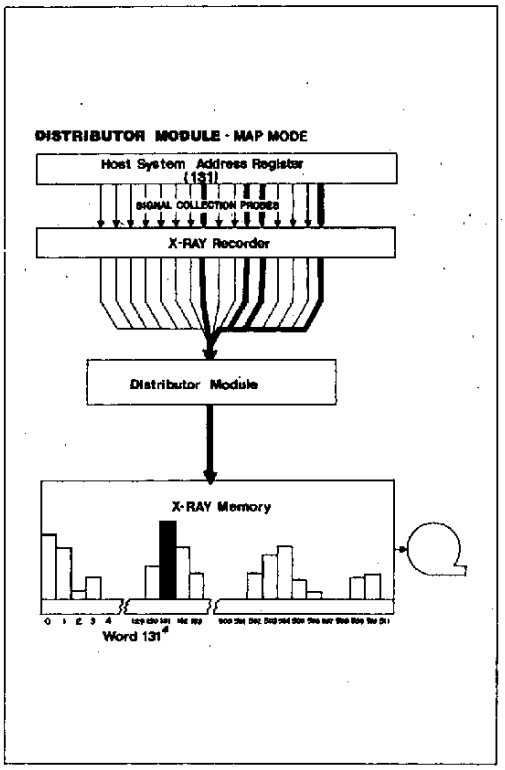


図3－18　E−Map modeの動作状態

c） T− Map mode

このモードは，パッチ・パネルからの信号の，タイム・デュレーションに従って，メモリ・モジュール内に度数分布をつくりあげるもの

である。

計測する信号時間間隔の許容範囲は，あらかじめセットしておけばチェック可能である。

このモードにおいても，連続的な測定を可能にするため 512 ワードの代替バッファを用意している。動作状態は図 3−18 と同様である。

⑤ リアルタイム・クロック

24 時間のリアルタイム・クロックを有し，パネルへの表示は時間と分であるが，Recorder への制御信号としては，1 μsec〜10 分までのパルスを発生することができる。

⑥ 出力用モジュール

出力用モジュールは，メモリ・モジュールから測定データを磁気テープに出力するものである。

磁気テープは 9 トラック／800 bpi，9 トラック／1600 bpi および 7 トラック／800 bpi の 3 種がセット可能で，パッチ・パネルからの信号によって日付時刻その他の制御情報が先頭に記録できるようになっている。1 レコードを記録している間も測定可能にするため，各モードに交替用のバッファが利用できるようになっている。

⑦ メモリ・モジュール

メモリ・モジュールとしては 2048 あるいは 4096 ワード（16 bits／w）の磁気コア・メモリがつき，そのアクセスに Read／modify／write，Read／restore，破壊読出しの 3 種のモードを設定している。

メモリの最大サイクル・タイムは 1.2 μs である。

⑧ その他

制御用パネルを図 3−19 に示す。

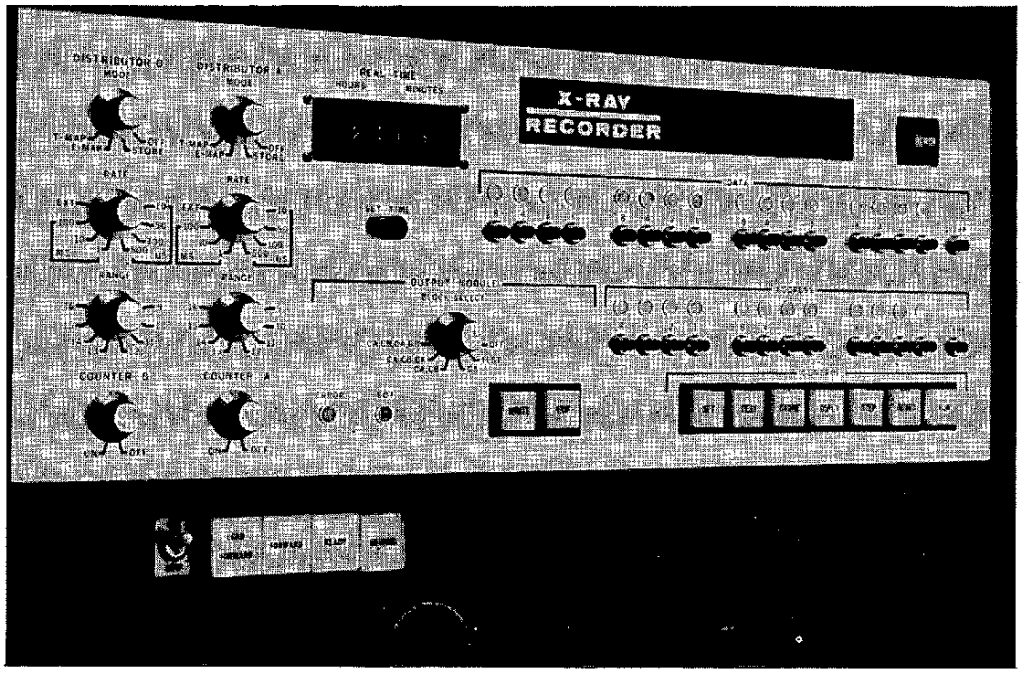


図3－19　前面パネル

(8)　解析用ソフトウェア

X-RAYシステムには，測定データの解析用ソフトウェアとしてanalyzer，Editor，Intercept moduleの3種を用意している。

Analyzer は測定した各装置，プログラム，それにデータの利用状況など，コンピュータ・システムのパフォーマンスを示すデータのサマリ・レポートを作成するプログラムである。このプログラムはANS-FORTRANで書かれており，Recorder で測定したシステムのアクティビティを適当なスコープで拡大したり，ある特定な時間だけに焦点をあてた解析レポートを作成でき，その為のパラメータがいくつか用意されている。

この，測定後の関連付けの機能により，基礎的な測定データで，かなりの程度まで深く分析できるようになる。また，入力としては生の測定データだけではなく，Editor や，Intercept module により加工したデ

ータも利用できるようになっている。

図3－20にAnalyzer の機能概要を示す。

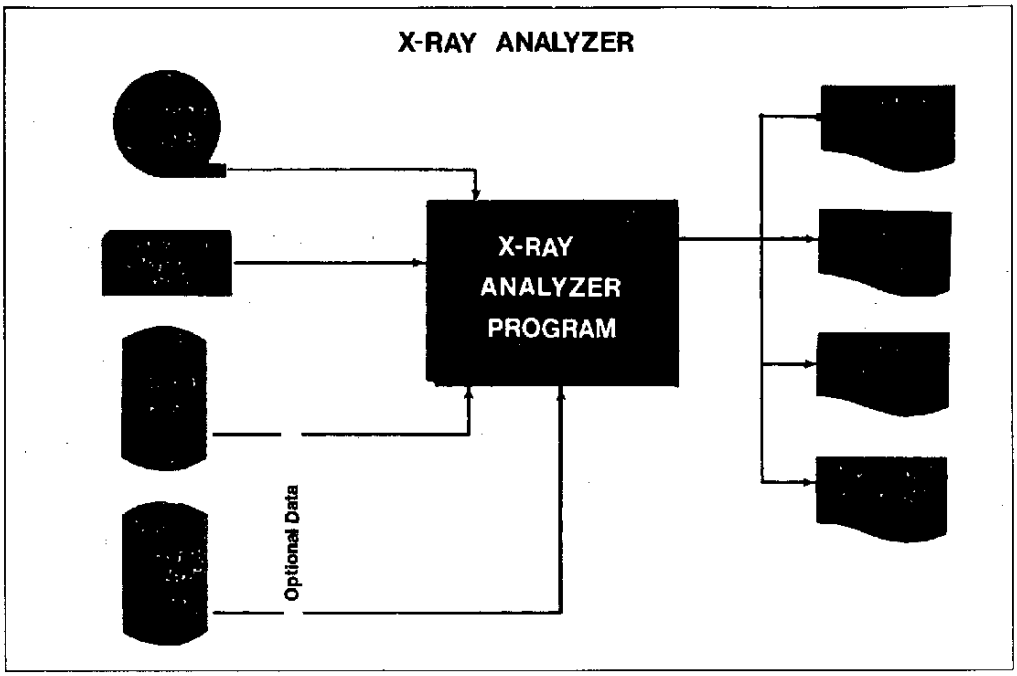


図3－20　X－RAY Analyzer の概要

Analyzer によって得られる各レポートは以下のようになっている。

① カウンタ・サマリ・レポート

② カウンタ・インターバル・レポート

③ Store mode のヒストグラム

④ 汎用ディストリビュータ・レポート

⑤ E－map スライス・ヒストグラム

⑥ 合成パラメータ・サマリ

⑦ プロブレム・プログラムの状態

⑧ 合成パラメータによるレポート

⑨ プログラム解析のヒストグラム

⑩ データ・ベースのボリューム・マップ

⑪ ボリューム・アクティビティ・レポート

実際に作成されたレポートの例の一部を図3−21から図3−24に示す。

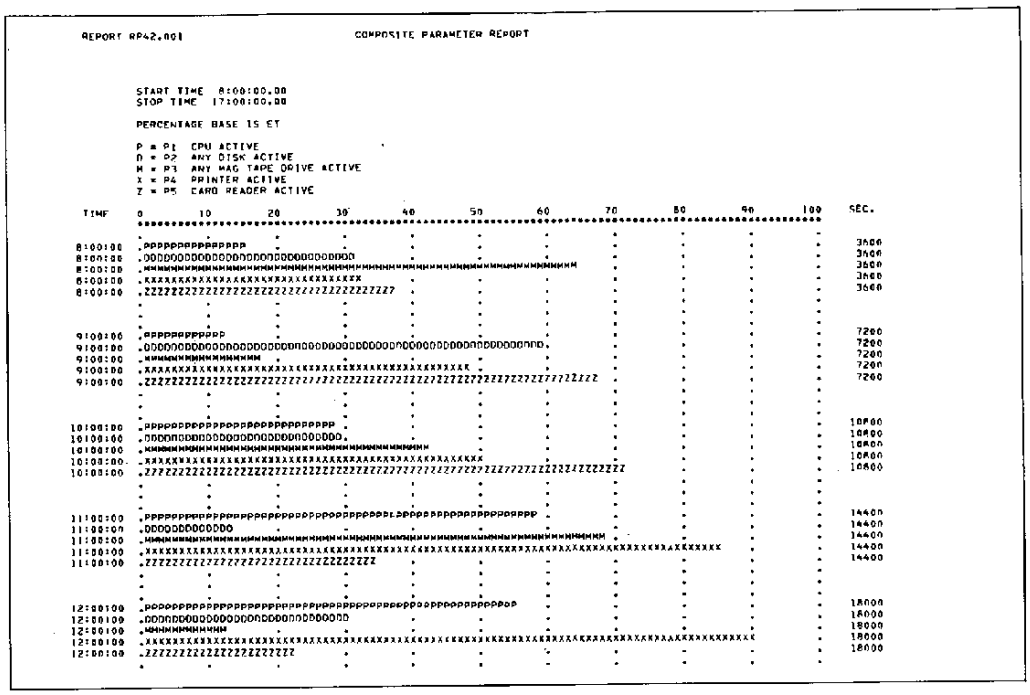

REPORT RP42.001
COMPOSITE PARAMETER REPORT
START TIME 8:00:00.00
STOP TIME 17:00:00.00
PERCENTAGE BASE IS ET
P = P1 CPU ACTIVE
D = P2 ANY DISK ACTIVE
M = P3 ANY MAG TAPE DRIVE ACTIVE
X = P4 PRINTER ACTIVE
Z = P5 CARD READER ACTIVE
TIME
SEC.

図 3 − 21

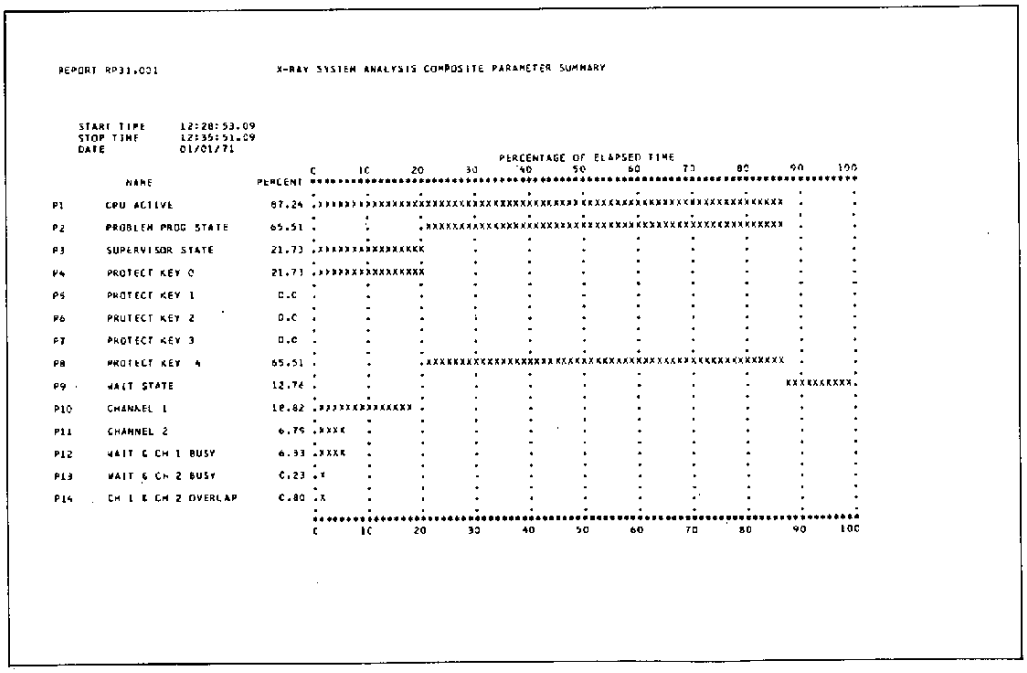

REPORT RP31.001
X-RAY SYSTEM ANALYSIS COMPOSITE PARAMETER SUMMARY
START TIME 12:28:53.09
STOP TIME 12:35:51.09
DATE 01/01/71
PERCENTAGE OF ELAPSED TIME
NAME
PERCENT
P1 CPU ACTIVE 87.24
P2 PROBLEM PROG STATE 65.51
P3 SUPERVISOR STATE 21.73
P4 PROTECT KEY 0 21.73
P5 PROTECT KEY 1 0.0
P6 PROTECT KEY 2 0.0
P7 PROTECT KEY 3 0.0
P8 PROTECT KEY 4 65.51
P9 WAIT STATE 12.76
P10 CHANAEL 1 10.82
P11 CHANNEL 2 0.79
P12 WAIT & CH 1 BUSY 0.33
P13 WAIT & CH 2 BUSY 0.23
P14 CH 1 & CH 2 OVERLAP 0.80

図 3 − 22

REPORT RP61.001　　　VOLUME ACTIVITY REPORT

REPORT START TIME 13:20:00.00
　　　STOP TIME 14:20:00.00
DATA　START TIME 13:20:00.00
　　　STOP TIME 14:20:00.00

VOLUME SERIAL = IPLVI8

DATA SET NAME	EXT NO	SEC NO	ADDRESS TRK-TRK	SEEK COUNT	PERCENT	PERCENT OF SEEKS 0 . . . 10 . . . 20 . . . 30 . . . 40
VTOC	0	1	0000-0019	31	0.12	.X
SYSCTLG	0	2	0020-0039	14	0.05	.X
SYS1.PROCLIB	0	3	0040-0059	32	0.12	.X
	0	3	0060-0079	10	0.04	.X
	0	3	0080-0099	0	0.0	.
	0	3	0100-0119	84	0.31	.X
SYS1.SYSJOBQE	0	4	0120-0139	0	0.0	.
	0	4	0140-0159	0	0.0	.
	0	4	0160-0179	0	0.0	.
	0	4	0180-0199	0	0.0	.
	0	4	0200-0219	0	0.0	.
SYS1.SVCLIB	0	5	0220-0239	5,657	21.14	.XXXXXXXXXXXXXXXXXXXXX
	0	5	0240-0259	1,909	7.13	.XXXXXXXX
	0	5	0260-0279	1,726	6.45	.XXXXXXX
	0	5	0280-0299	398	1.49	.X
	0	5	0300-0319	0	0.0	.
	0	5	0320-0339	34	0.13	.X
	0	5	0340-0359	12	0.04	.X
	0	5	0360-0379	0	0.0	.
	0	5	0380-0399	0	0.0	.

図 3 − 23

REPORT RP51.001　　　X-RAY PROGRAM ANALYSIS HISTOGRAM

REPORT START TIME 12:30:00.00
　　　STOP TIME 12:31:25.00
DATA　START TIME 12:30:00.00
　　　STOP TIME 12:31:25.00

PROGRAM = UPDAT4

NAME	SEQ	ADDRESS	SAMPLES	PERCENT	0 . . . 10 . . . 20 . . . 30 . . . 40
MAIN	1	00000-003FF	0	0.0	.
	1	00400-007FF	0	0.0	.
	1	00800-00BFF	0	0.0	.
	1	00C00-00FFF	0	0.0	.
	1	01000-013FF	0	0.0	.
ACDCCM	2	01400-017FF	0	0.0	.
COMMON	3	01800-01BFF	0	0.0	.
COPYC	4	01C00-01FFF	0	0.0	.
COPYI	6	02000-023FF	0	0.0	.
COPYL	7	02400-027FF	0	0.0	.
	7	02800-02BFF	0	0.0	.
COPYO	8	02C00-02FFF	44,135	2.91	.XXXX
DSTORE	10	03000-033FF	903	0.06	.X
ERROR	11	03400-037FF	0	0.0	.
GETIN	12	03800-03BFF	0	0.0	.
	12	03C00-03FFF	6,284	0.41	.X
GETOLD	13	04000-043FF	62,746	4.14	.XXXXXX
GETI	14	04400-047FF	567,083	37.43	.XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
ICONV	15	04800-04BFF	13,979	0.92	.X
INCLUD	16	04C00-04FFF	12,703	0.84	.X

図 3 − 24

B. *Micro SUM

(1) Micro SUM の概要

Micro SUM は，米国のComputer Synektics社で開発した非常にコンパクトで安価なハードウェア・モニタである。本体のサイズは10" × 12" × 15"インチと小型で，重量も16ポンド少々とコンパクトになっている。Synectics社は，高級ハードウェア・モニタSUMを開発している会社であるが，このMicro SUMは，SUMの経験を生かした超小型版と言える。

価格も基本構成のもので$ 2,995と，他のハードウェア・モニタに比べたら桁違いに安くなっている。

Micro SUM の外観を図3−25に示す。

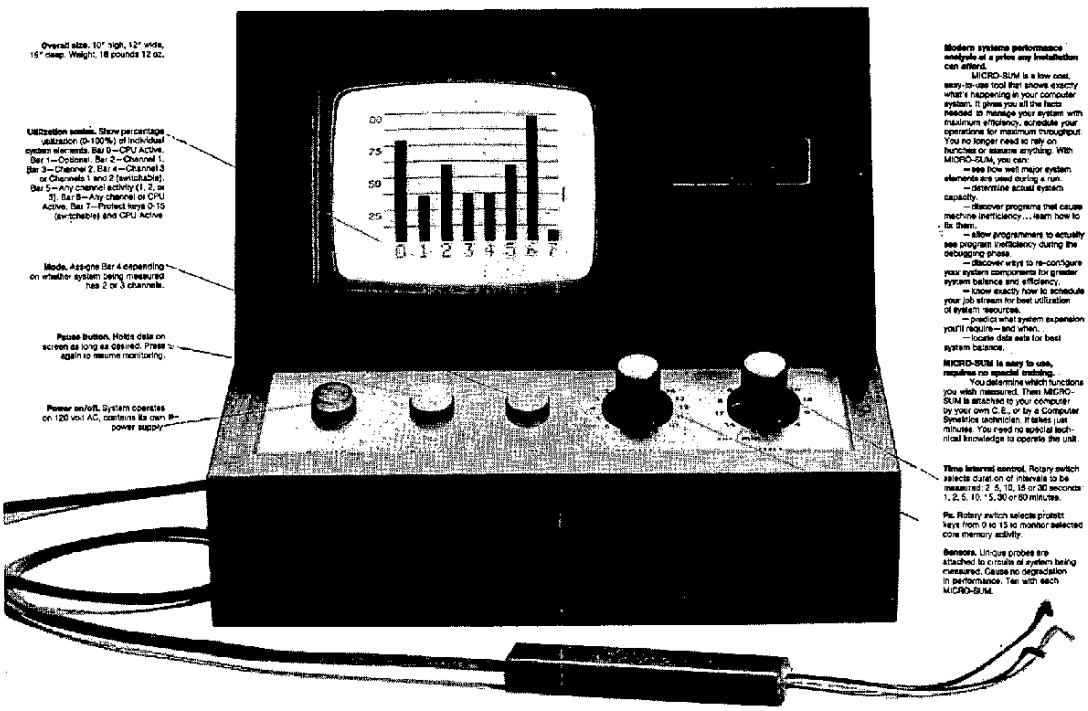


図3 − 25　Micro SUMの外観

*Micro SUMについては，Computer Synektics 社よりパンフレットを頂き図を引用させて頂いた。

次にMicro SUM 特性の諸元を以下に示す。

プローブの数は10個までで，ケーブルの長さは15フィートまでである。

測定のリゾルーションは25nsまでで，測定時間は2，5，10，15，30秒，1，2，5，10，15，30，60分のきざみをロータリ・スィッチで切換えることができるようになっている。また，測定結果は，パネル前面に直接，グラフで表示されるようになっており，8種の測定データをパーセント表示で同時に表わすことができる。表示結果は，ダイナミックに変化するが，プッシュ・ボタンにより一時静止させたり，リスタートさせたりすることができる。以上がMicro SUM model-1 基本構成の概要である。

さらに，オプションとして以下のような拡張が可能である。

① プローブの数が20個

② ロジック・パネルの追加

③ CRT display にディジタル・クロックの組込み

④ リモート・ディスプレイの追加

⑤ 測定データをカセット・テープに自動記録

⑥ プリンタの取付け

⑦ 記録された測定データの編集と解析を用うためのソフトウェア SUMEDIT，SUMDAP

(2) Micro SUMの利用例

以下の5例は，いずれもシステムのボトルネックをMicro SUM によって観測した例である。

以下のグラフにおける各Bar は次のような測定量を表示している。

Bar 0 ：CPUアクティブ

Bar 1 ：オプションであり，本図では用途不明

Bar 2 ：チャネル1

Bar 3 ：チャネル2

Bar 4 ：チャネル1 "AND" チャネル2

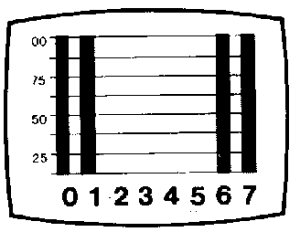
Bar５：チャネル１"OR"チャネル２

Bar６：チャネル"OR"CPU

Bar７：プロテクト・キー "AND" CPU

① CPUの利用度が異常に高く，他のリソースを全く利用していない例

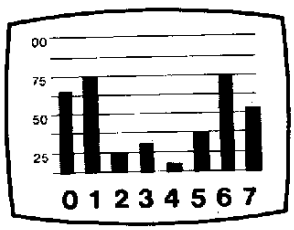
Bar 0はCPUのアクティビティが100%であることを示し，Bor２，３が０であるのはCPUがごく限られたリソースにのみ用いられていることを示している。原因としては極端なCPUバウンドのプログラム実行中か，割込み処理内でループすることなどが考えられる。



Abnormally high CPU activity—no other resource active (bars 0, 2, 3)

② CPUの利用度は高いがチャネルの利用が低い例

このグラフは，かなりコンピュート・バウンドなシステム状態を示しており，このような例はジョブ・ストリームの構成がうまく行ってない時に起る。



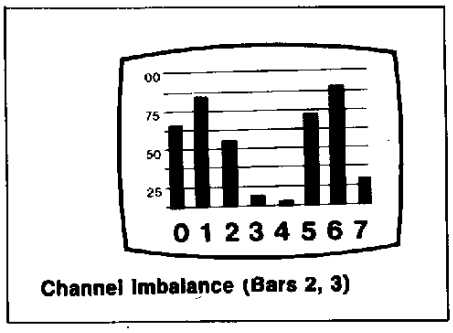
High CPU Activity—Low Channel Activity (Bars 0, 2, 3)

③ CPUの利用度が低く，チャネルの利用が多い例

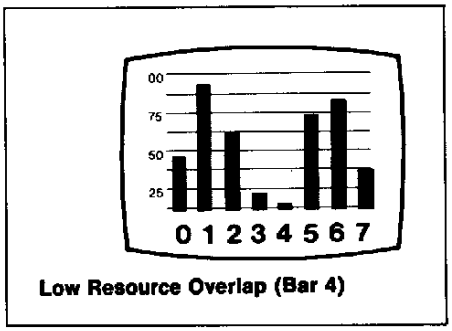
この例は，②とは逆にI／Oバウンドのシステム状態になっていることを示している。

Low CPU Activity—High Channel Activity (Bars 0-5)

④　チャネルのバランス不良の例



⑤　リソースのオーバラップ利用が低い例



3.4 ソフトウェア・モニタ

3.4.1 ソフトウェア・モニタの種類

商用のソフトウェア・モニタとしては，第1節の表3－4（P　　）に示すように，各種のシステムが提供されている。

このうち，M-TEST，STAGEⅡ，COBOL Optimiger，FORMAXは，コンパイラ・レベルの言語で書かれたプログラムをオプティマイズするためのもので，利用目的から言えば，いわゆるオプティマイザというタイプに属する。

次に，利用者の，あるプログラムの動きに着目し，その実行状態をモニタリングするものとして，SMSのPPE，PROGLOOK，LEAPがある。これらはいずれも，アセンブラからPL／1などまで，記述言語によらずにプログラム効率を解析できるようになっている。これも，利用目的から言えば，ある単一のプログラムのオプティマイザになる。

次に，システム全体のパフォーマンスをモニタリングするためのものとしてSMSのCUE，SUPERMONがある。

また，これらのソフトウェア・モニタとは少しタイプは異なるが，ユーザ・プログラムの実行をトレースしてその結果を分析するものとしてIBMのAMAPがあげられる。本章ではこれを，シミュレータの節で説明しているが，利用範囲を限れば，AMAPもソフトウェア・モニタとしてとらえることができる。

今まで述べたものはすべて，ユーザ・プログラムの一つとして実行され，他のプログラムあるいはシステムをモニタリングするものであるが，OS組込みのモニタリング・システムとしてIBMのSMFがある。これは，本来，アカウンティング情報を得るためのものであろうが，第2節でも述べたように，利用のしかたによっては，パフォーマンス向上に有用であり，レベルは違うが，やはり，ソフトウェア・モニタの一種になる。

次にこれらのソフトウェア・モニタを第2章で述べたイベント・タイプのものとサンプリング・タイプのものとに分けると，AMAP，SMFがイベント・

タイプで，他はすべてサンプリング・タイプになっている。

次項にソフトウェア・モニタの実例としてSMSとSTAGEⅡ，FORMAX，COBOL OPTIMIZERをあげ，機能説明に変える。

3.4.2. ソフトウェア・モニタの実例

A. *SMS

(1) 概　要

SMS（Systems Management Software）は，米国のBool & Babbage社が提供している，システム運用効率向上をはかるためのソフトウェアである。

SMSでは，実働中のオペレーティング・システムからサンプリング手法によって必要なデータを抽出し，統計的な処理を加えて，各種のレポートを作成するようになっている。

SMSは発表以来5年を経過し，その間ユーザからのフィード・バックを得て改良が加えられて今日に至っており，現在約300社のIBM大型機ユーザで利用されていると言われている。

SMSは以下の3システムから構成されている。

- PPE（Problem Program Evaluator）

  一本のProblem Programを測定対象として，実行時における効率を定量的に測定する。

- CUE（Configuration Usage Evaluator）

  CPUを中心として，その周辺機器およびオペレーティング・システムの実働状況を，あるジョブ・ミックスの実行中に定量的に測定する。

- DSO（Data Set Optimizer）

＊SMSについてはBoole & Balbage社より資料の提供を受け，さらにSMSの日本代理店である東京システム技研㈱松本氏の資料を引用させて頂いた。

可動ヘッドを持つ磁気媒体の１ボリュームを測定対象として，その利用状況を定量的に測定し，更に同じ利用状況に対するデータ・セット割付けの改善案を提示する。

(2) 共通の特性

① データの収集

内部インタバルタイマーによる等間隔のインタラプションをベースとしたサンプリング方式をとっている。サンプリング・レイトは，16.7ｍｓまたは20.0ｍｓの整数倍で制御パラメーターにより決定される。

② プログラム形式

各製品はそれぞれ，ＥｘｔｒａｃｔｏｒおよびＡｎａｌｙｚｅｒ と呼ばれる２つのプログラムから成っている。Ｅｘｔｒａｃｔｏｒはインタラプション・オリエンテッドでサンプルデータを抽出し，Ｅｘｔｒａｃｔｏｒデータセットに書き出す。Ｅｘｔｒａｃｔｏｒデータセットは磁気テープ／ディスクのどちらでもよい。Ａｎａｌｙｚｅｒ はこのデータセットを１個または複数個を組み合わせて，各レポートを作成する。各プログラムは通常のジョブまたはジョブステップとして実行され，それぞれの測定計画に従って，必要なオプションの選択とパラメーターの指定のための制御カードが使用される。

③ レポート形式

ラインプリンターから出力されるレポートは，それぞれがユーザーの"とり得る手段"への橋渡しを目的とした具体的なサブレポートから成立っている。サブレポートはユーザーの測定計画に従って，制御カードによって選択されるが，全体として重点的な効率化の手順に便利なように設計されており，また，記述的な情報と数量的な情報とは対比がとりやすいことがこの測定方式の特長となっている。

④ 特に考慮のはらわれているポイント

1) サンプリングの為のオーバヘッド時間を少くすること。

2) Ｅｘｔｒａｃｔｏｒの占有容量による影響を少くすること

3）サンプリングのRandomness

4）Extractorデータセットへ書き出しの為のI／O競合をなくすこと

5）InterruptionのPriorityとProtection Keyが必要充分の条件であること

6）実働状態に対する制約条件が少ないこと

7）実行時，パラメーター指定などが容易なこと

8）レポート解読が容易なこと

(3) 各製品のレポート内容と利用分野

① PPEレポート

1）Extractor Data Set(s)の概要

2）サンプル範囲内外の消費時間分布

3）データセットに起因するWAIT時間分布

4）モジュールおよびセグメントごとの消費時間分布

5）詳細分析指定範囲内外の消費時間分布

6）〃　〃　内のデータセットに起因するWAIT時間

7）〃　〃　内のWAIT発生アドレスとその消費時間

8）〃　〃　内から呼んだ割込可能なSVC消費時間

9）単位アドレス間隔ごとのCPU消費時間（ヒストグラム）

10）特に指定したアドレス範囲のCPU消費時間

PPEのレポート例として3（図3−26），4（図3−27），9（図3−28）の例を示す。

```
DISTRIBUTION OF DSOW WAIT

   DATA SET
     NAME                                  PERCENT OF ACTIVITY

   JOBLIB                                          0.0
   SYSOUT                                          0.0
   SYSIN                                           0.0
   UNBLKED                                        22.73
   BLKED                                           2.37
                                                  ------
   TOTAL                                          25.10

                                                * * * * *
```

図３－26　データ・セットごとの Wait 時間の割合

```
MODULE MAP

   MODULE     FIRST BYTE   LAST BYTE     PERCENT OF      MODULES WITH      MODULES FOR WHICH
    NAME       ADDRESS      ADDRESS      RUN TIME         OVERLAYS       REPORTS ARE PROVIDED

   COBLTEST    001820       002B38         61.55                              X
 * IGG019CC    023DA8       02BE68          2.83
   IGG019AQ    02BC10       02BC88         34.84
   IGG019AA    02BB90       02BBF8          0.78
   IGG019CF    02BA48       02BB48          0.00

                                          * * * * *
```

図３－27　モジュールごとの消費時間の割合

```
   CODE EXECUTION FREQUENCY FOR EACH INTERVAL (EXCLUDING DSOW WAIT)

   STARTING    ENDING    INTERVAL CUMULATIVE HISTOGRAM - % OF TIME (EACH * = 0.5 %)
   LOCATION    LOCATION  PERCENT   PERCENT   .0      4.0      8.0      12.0     20.0

  .000000      00061F      0.00      0.00    -
   000620      00068F      7.39      7.39    -***************
   000690      0006FF      0.0       7.39    -
   000700      00076F      1.56      8.95    -***
   000770      0007DF     15.59     24.54    -********************************
   0007E0      00084F     13.94     38.48    -****************************
   000850      0008BF     12.58     51.06    -*************************
   0008C0      00092F      2.39     53.45    -*****
   000930      00099F       .41     53.86    -
  .0009A0      000B5F      0.0      53.86    -
   000B60      000BCF      1.28     55.14    -**
   000BD0      000C3F      1.84     56.98    -***
   000C40      000CAF      3.45     60.43    -*******
   000CB0      000D1F       .30     60.73    -
   000D20      000D8F       .04     60.77    -
   000D90      000DFF       .13     60.90    -
   000E00      000E6F       .24     61.14    -
   000E70      000EDF       .23     61.37    -
   000EE0      000F4F       .18     61.55    -
   000F50      001318      0.0      61.55    -
```

図３－28　単位アドレス間隔ごとのCPU消費時間

PPEの利用

1）プロブレム・プログラムの実行時間短縮とスループットの向上

2）プロブレム・プログラムの実行性向把握（CPU／IO Bounds）

3）プログラミング標準手法の効果確認　　等

② CUEレポート

1）CPUとCHANNELの各利用率とその相関関係

2）各DEVICEの利用率とその平均的な待行列実績

3）競合的な入出力操作要求実績

4）各可動ヘッド機番のヘッド移動実績

5）SVCトランジェント領域の利用実績

6）トランジェント型SVCの利用実績

7）Extractor Data Set(s)概要

CUEのレポート例として1（図3−29），2（図3−30），6（図3−31）の例を示す。

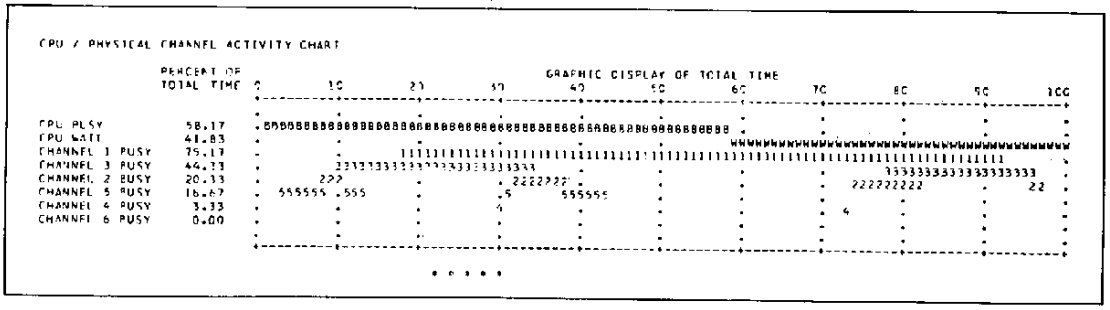
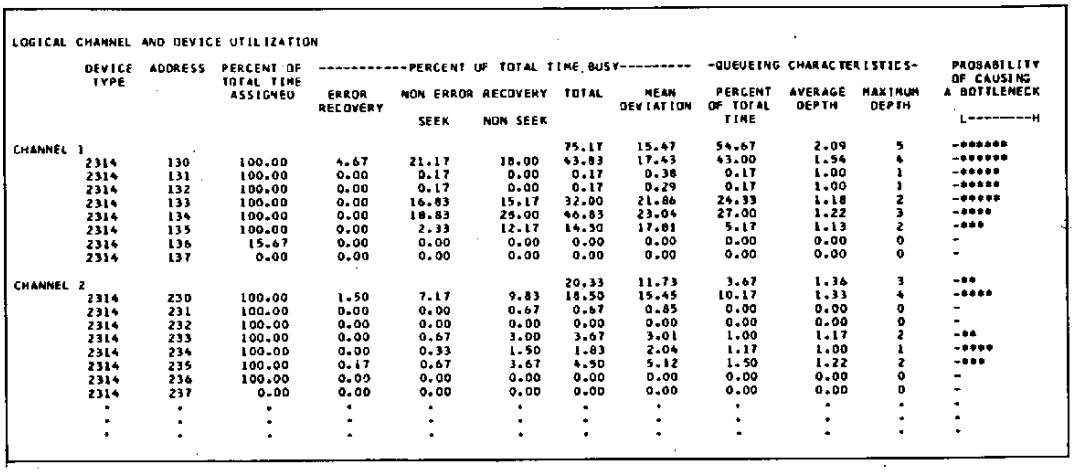


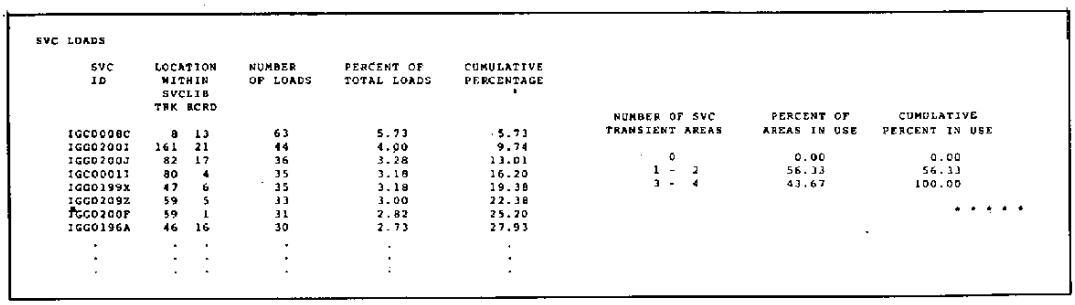
図3－29　CPUとチャネルの各利用率



LOGICAL CHANNEL AND DEVICE UTILIZATION

DEVICE TYPE	ADDRESS	PERCENT OF TOTAL TIME ASSIGNED	PERCENT OF TOTAL TIME BUSY					QUEUEING CHARACTERISTICS			PROBABILITY OF CAUSING A BOTTLENECK
			ERROR RECOVERY	NON ERROR SEEK	RECOVERY NON SEEK	TOTAL	MEAN DEVIATION	PERCENT OF TOTAL TIME	AVERAGE DEPTH	MAXIMUM DEPTH	L---------H
CHANNEL 1						75.17	15.47	54.67	2.09	5	-******
2314	130	100.00	4.67	21.17	18.00	43.83	17.43	43.00	1.56	6	-*******
2314	131	100.00	0.00	0.17	0.00	0.17	0.38	0.17	1.00	1	-*****
2314	132	100.00	0.00	0.17	0.00	0.17	0.29	0.17	1.00	1	-*****
2314	133	100.00	0.00	16.83	15.17	32.00	21.86	24.33	1.18	2	-******
2314	134	100.00	0.00	18.83	28.00	46.83	23.04	27.00	1.22	3	-*****
2314	135	100.00	0.00	2.33	12.17	14.50	17.81	5.17	1.13	2	-****
2314	136	15.67	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0	-
2314	137	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0	-
CHANNEL 2						20.33	11.73	3.67	1.36	3	-**
2314	230	100.00	1.50	7.17	9.83	18.50	15.45	10.17	1.33	4	-*****
2314	231	100.00	0.00	0.00	0.67	0.67	0.85	0.00	0.00	0	-
2314	232	100.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0	-
2314	233	100.00	0.00	0.67	3.00	3.67	3.01	1.00	1.17	2	-**
2314	234	100.00	0.00	0.33	1.50	1.83	2.04	1.17	1.00	1	-*****
2314	235	100.00	0.17	0.67	3.67	4.50	5.12	1.50	1.22	2	-****
2314	236	100.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0	-
2314	237	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0	-
.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.

図３－30　各デバイスの利用率と待行列



SVC LOADS

SVC ID	LOCATION WITHIN SVCLIB TRK RCRD	NUMBER OF LOADS	PERCENT OF TOTAL LOADS	CUMULATIVE PERCENTAGE
IGC0008C	8 13	63	5.73	5.73
IGG02001	161 21	44	4.00	9.74
IGG0700J	82 17	36	3.28	13.01
IGC0001I	80 4	35	3.18	16.20
IGG01999X	47 6	35	3.18	19.38
IGG0209Z	59 5	33	3.00	22.38
IGG0200F	59 1	31	2.82	25.20
IGG0196A	46 16	30	2.73	27.93
.	. .	.	.	.
.	. .	.	.	.
.	. .	.	.	.

NUMBER OF SVC TRANSIENT AREAS	PERCENT OF AREAS IN USE	CUMULATIVE PERCENT IN USE
0	0.00	0.00
1 - 2	56.33	56.33
3 - 4	43.67	100.00

* * * * *

図３－31　トランジェントSVC利用実績

CUEの利用

1）資源の競合，待行列などのかくされたスループット・ロスの発見

2）CPU WAIT の直接的な原因の把握

3）オペレーティング・システム機能の効果的な利用方法の発見

4）ジョブ・スケジューリングの改善資料

5）機器構成，OS／SYSGENの改善または増強時の資料　　等

③　DSOレポート

1）Extractor Data Set(s)概要

2）被測定ヴォリューム概要

3）被測定ヴォリューム内容のMapping

4）ヘッド移動位置・回数・時間の実績

5）データ・セット再配置改善案と改善率予想値

DSOレポートの例を4（図3−32），5（図3−33）に示す。

DATA SET HEAD MOVEMENT ON VOLUME BOOL7Ø

DATA SET PAIRS	NUMBER OF TRAVERSALS BETWERN DATA SETS	HEAD MOVEMENT TIME	PERCENTAGE OF HEAD MOVEMENT TIME	AVERAGE HEAD MOVEMENT TIME
PBFILE (01) - PBFILE (02)	108127	8758287 MS	49.00	81.01 MS
PPFILE (01) - PPFILE (02)	86920	5997480 MS	34.50	69.00 MS
PBFILE (01) - PBFILE (01)	19529	637817 MS	8.18	32.66 MS
	.	.	.	.
	.	.	.	.
	.	.	.	.
	238680	17378290 MS	100.00	72.81 MS

図3－32　ヘッド移動位置・回数・時間の実績

REORGANIZATION OF VOLUME BOOL7Ø

DATA SET NAME	EXTENT	DATA SET BOUNDARIES CC	TT	CC	TT
VTOC	01	0	1	0	10
*** FREE SPACE	01	0	11	0	19
PBFILE	02	1	0	1	19
PBFILE	01	2	0	111	19
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.

TOTAL HEAD MOVEMENT TIME BEFORE REORGANIZATION　17378290 MS
APPROXIMATE HEAD MOVEMENT TIME AFTER REORGANIZATION　12979418 MS
AVERAGE HEAD MOVEMENT TIME AFTER REORGANIZATION　54.36 MS

PERCENTAGE IMPROVEMENT RESULTING FROM REORGANIZATION　25.30

図3－33　データ・セット再配置改善案

DSOの利用

1）ディスクパック再編成と無駄なスペース発見の資料

2）DASD 増設の資料

(4) PPEの実験結果

PPEを実際に利用した例として米国のThe Guardian life Insurance 社の実験結果を以下に示す。

表3－6 *Guardian社におけるPPE実測例

	UPDATE プログラム改善点 1. EXAMIN 文を1つ削除 2. 添字をバイナリにした 3. あるルーチンを必要な時だけ呼ぶ	BILLING プログラム改善点 1. EXAMIN 文2つの変更 2. 添字をバイナリにした
CPU 時間	35分3秒	33分52秒
	20分1秒 （ 42.8％）	19分49秒 （ 41.5％）

B. **STAGEⅡ

(1) 概 要

STAGEⅡは，米国のTesdata社が提供している，COBOLソース・プログラムのオプティマイザである。

STAGEⅡへの入力は，利用者が作成したCOBOLのソース・プログラムであり，そのプログラムを分析して，実行時間の減少，コア・サイズの縮少など，プログラムの最適化に役立つような診断結果のリストを出力するものである。利用者の指定により，修正すべき部分のカード・イメージ修正ファイルが出力されるので，原プログラムやライブラリを編集してオプティマイズされたプログラムをつくるのは比較的簡単である。また，イ

＊本結果は米国調査でThe Guardian life Insurance 社を訪問した折入手した資料による。

＊＊STAGEⅡについてはTesdata 社よりパンフレットの提供を受け，その内容をサマライズしたものである。

ンストレーション特有の標準的プログラミング規則をSTAGEⅡに与えておくことも可能であり，その標準に適合しないプログラムに対しては警告メッセージを出すこともできる。

(2) オプティマイズに要する時間

STAGEⅡにより，一つのプログラムを分析し，修正カードを出力するために要する時間は，通常のコンパイル時間の約⅓である。出力された診断メッセージと修正カードにより，カードの修正をおこない，さらに，プログラム構造上の最適化をおこなうための参考情報が出力されているので，プログラム全体の検討を加え，変更すべきかどうかを決定し，オプティマイズしたプログラムのテストに至るまでの時間は，プログラムの大きさにもよるが通常 10分～4時間程であるということである。

(3) STAGEⅡの利点

以下にTesdata社が言明しているSTAGEⅡの利点をあげる。

- COBOLプログラムの約70％は，高度の最適化が必要であり，それらに対して実行時間で平均 25％縮少させることができる。
- コア・サイズの縮少ができる。
- プログラミングの教育に役立つ。
- プログラミングの標準化に役立つ。
- プログラミングの技術評価に役立つ。

C. *FORMAX

(1) 概　要

FORMAX（FORTRAN Maximizer）は米国のComputer Synectics社が提供しているFORTRANプログラムのオプティマイザである。

過去の経験によれば，FORTRANプログラムの多くは，プログラム中

＊FORMAXについてはComputer Synectics社よりパンフレットの提供を受け，その内容をサマライズしたものである。

のごく小さい部分が，実行時間の大部分を占めていると言われており，FORMAXはこのクリティカルな部分を自動的に指摘する機能を有したオプティマイザである。FORMAXは，FORTRANプログラムの各ソース・ステートメント単位あるいは，ソースのあるブロック単位に，使用頻度および，実行に要する時間を示してくれ，FORTRANプログラムのどの部分が，実行時間に大きく影響しているかを示してくれるもので，この部分を修正すれば，実行時間を約半分に縮めることが可能であろうということである。

(2) FORMAXによるレポート

FORMAXにより分析した結果は，次の5種のレポートにまとめて出力される。

- 使用頻度
- 実行時間のサマリ
- プログラム・アクティビティのサマリ
- サブルーチンのサマリ
- ステートメントの種類に関するサマリ

これらのデータはいずれもFORTRANソース・ステートメントに対応させて累積されており，クロス・リファレンスや，リンケージ・マップなどを参照する必要は，まったくないようになっている。また，各サマリは，大きい順に並べて印刷されるので，プログラムのネックは，一目で発見できるようになっていると言う。

現在，FORMAXはIBM 360/OSのもとで利用可能で，値段は$1,850である。

D. *COBOL OPTIMIZER

(1) 概　要

OPTIMIZERは，米国のCAPEX社が提供している，COBOLプログラムのオブジェクト・コード・レベルでのオプティマイザである。

今までに述べたSTAGEⅡとFORMAXは，ソース・ステートメント上でのオプティマイズ情報を出力するものである。これはコンパイラの上手な使い方，あるいはプログラミング・テクニックの拙劣さをカバーするのに有用となるわけだが，OPTIMIZER では，コンパイラのつくり出すオブジェクト効率の悪い点をオプティマイズするアプローチをとっている点で，他のオプティマイザとは異なっている。オブジェクト・コードのオプティマイズは，主に次の3点について自動的におこなわれる。

- レジスタ・ロードの最小化
- アドレス定数の最小化
- PERFORMのリンケージ・コードの最小化

上記のようなオプティマイズをおこなうために，プログラムのすべてのパスおよびループを解析し，汎用レジスタの使用，およびプロシーデュア・コントロール用命令に対するグローバルなストラトジーを決め，そのコードを変更する方法をとっている。

この方法により，プログラムの多くは，プロシーデュア部分のコーディングに要するコア・スペースを25％～35％減少し，データ部分も含めたプログラム全体のコア・スペースは8％～20％は減少するということである。

このようにしてオプティマイズされたプログラムは，走行命令の数が減少しているわけだから，当然実行時間は速くなり，CPU時間で8～15％

＊COBOL OPTIMIZERについては，CAPEX 社の日本代理店である㈱アシスト社より資料の提供を受け，図表などを引用させて頂いた。

は減少すると言っている。これにより，システム・オペレーションに与える大きな利点は，一つ一つのプログラムの主記憶占有量縮少と実行時間のスピード・アップにより，スループットがあがることであろう。またCPUバウンドのプログラムの場合はこれだけで，自動的に効率が向上するが，I／Oバウンドのプログラムの場合は節約したコア・スペースをバッファ・エリアの拡張に割り当てるようなオプション機能も有している。

OPTIMIZERを利用することにより直接得られる利点としては以下のようなものがある。

- オブジェクト・コードの品質が向上する。
- CPU時間と主記憶の占有量が減少する。
- パーティション・サイズを減らすことができる。
- I／Oバッファリングの効率が向上する。

また，プログラムに対する間接的な影響としては，バッファ容量が増えることにより実行時間が速くなり，オーバレイの数を減らすことができ，標準的なリジョン部分で実行可能になる。

システム全体にとっても，マルチ・プログラミングの効果をより以上向上することができ，大きなプログラムによるスケジュール上のボトルネックが解消され，OSの重要なモジュールをレジデントにすることも可能になる。

(2) OPTIMIZERの動作

OPTIMIZERへの入力は，COBOLコンパイラがつくり出すオブジェクト・モジュールと，リスティング用のファイルである。出力はオプティマイズした新しいオブジェクト・モジュールと新しいリストである。

図3－34にOPTIMIZERの動作図を示す。

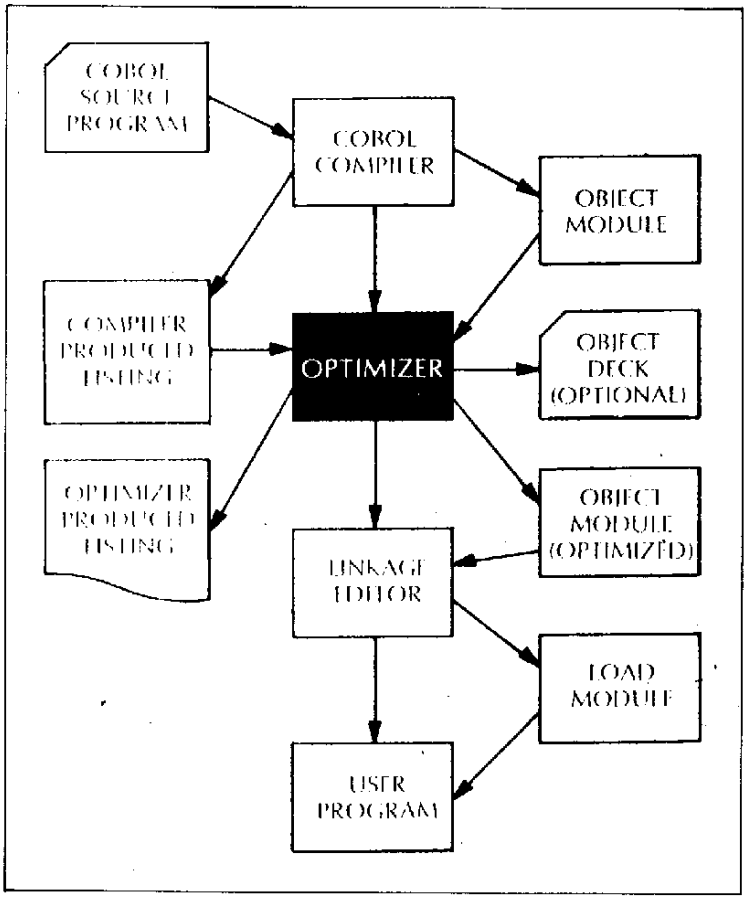


図3－34　OPTIMIZER動作図

(3)　オプティマイズ結果の例

オブジェクト・コードが実際にオプティマイズされた例として，オプティマイズ前のオブジェクト・コード（図3−35），オプティマイズ後のオブジェクト・コード（図3−36）を示す。図をみれば明らかなように，MOVEで10バイトが4バイトに，PERFORMでは30 バイトが4バイトに減少している。

```
47        *PND
                  0008B0                          PN=04   EQU   *
48        MOVE    0008B0  58 E0 D 0F0                     L     14,0F0(0,13)          BLI=14
                  0008B4  D2 00 E 004 C 171               MVC   004(1,14),171(12)     DNM=2-115     LIT+1
49        PERFORM 0008BA  58 00 D 218                     L     0,218(0,13)           VN=08
                  0008BE  50 00 D 1B0                     ST    0,1B0(0,13)           PSV=8
                  0008C2  56 00 C 0D0                     L     0,0D0(0,12)           GN=09
                  0008C6  50 00 D 218                     ST    0,218(0,13)           VN=08
                  0008CA  58 10 C 0A0                     L     1,0A0(0,12)           PN=040
                  0008CE  07 F1                           BCR   15,1
                  0008D0                          GN=09   EQU   *
                  0008D0  58 00 D 1B0                     L     0,1B0(0,13)           PSV=8
                  0008D4  50 00 D 218                     ST    0,218(0,13)           VN=08
```

図3－35　オプティマイズ前のオブジェクト

```
                  0006A4                          PN=04   EQU   *
00047 PND.
00048     MOVE 0 TO COND1.
                  0006A4  D2 00 2 004 D 249               MVC   004(1,2),249(13)      DNM=2-115     LIT+1
00049     PERFORM PN36 THRU PN37.
                  0006AA  45 F0 3 692                     BAL   15,692(0,3)           GN=062
```

図3－35　オプティマイズしたオブジェクト

表3－7は，米国のThe Guardian life Insurance 社で，このOPTIMIZERを利用して実測した結果の例である。

＊表3－7　Cuardian社によるOPTIMIZER測定結果

プログラム	コア占有量		実行時間	
	前	後	前	後
1	55592	46034	85 min	52 min
2	104838	78202	—	—
3	76768	56520	13.55"	12.40"
4	20804	14388	—	—
5	76590	65210	33	20
6	50888	39898	35	20
7	23858	21006	1.31"	1.10"
8	13760	12300	1.58"	1.40"
9	10488	10132	1.14"	0.43"
10	43736	23252	38.02"	35.24"

＊本結果は米国調査でThe Guardian life Insurance 社を訪問した折入手した資料による。

前表からもわかるように，コア・スペースで平均23％，実行時間で平均 32 ％の減少がみられる。

3.5 専用シミュレータ

3.5.1 専用シミュレータの種類

コンピュータ・システム評価用の商用シミュレータとしては，第1節の表3—4（P　　）のように各種のシステムが提供されている。

このうちSCERTおよびCASEは，各メーカのハードウェア・ソフトウェアの各仕様，それに経験的なデータや実験式など数多くの内容を含むファクタ・ライブラリを備えているのがその特色であろう。この種のシミュレータの場合は，利用者が，システム環境，ファイル仕様，ワーク・ロード特性などを入力することにより，ワーク・ロードの特性と処理仕様に対応するモデルができあがる。それに加えてハードウェア・ソフトウェア名称などを指示すれば，ファクタ・ライブラリからの抽出により，シミュレートすべきシステム構成のモデルができあがる。

これらのモデルをもとにシミュレートがおこなわれるわけだが，シミュレーションにあたっては，要求された評価項目に適用すべき最適な実験式を自動的に選択し，まず，単一のワーク・ロードごとにスループットを求める。次に，モデル化された全ワーク・ロード・モデルの確率的特性にもとづいて，単一の結果を並列に積み重ねることにより結果を得るようになっている。

次に，CSSの場合は，利用者がシステム・モデルをCSS言語を用いて，かなり詳細に記述し，イベント・オリエンティッドにシミュレートするようにっている。このような，いわゆる専用言語に近いタイプのシミュレータの場合は，モデルの記述に非常に柔軟性がある一方，利用者レベルであっても，たとえばOSの動作内容など，かなり細かくシステムの動作を知らないとモデル化できないという欠点がある。したがって，このような高等技術を必要とするシミュレータは，OSの設計および改良などのレベルで多く利用されている。

CSSに似たタイプのシミュレータとしてSAMがあるが，SAMの場合はある程度経験的なアプローチが加わっていて，IBM 360/370とかUNIVAC 1108などに関するモデル・ライブラリを利用できるようになっている。もち

ろんCSSにおいても，ごく限られた基本的な部分モデルは利用できるようにはなっているが，この種の，イベント・オリエンティッドなシミュレータの場合は，精度とシミュレートに要する時間のかねあいが問題になる。

次に，ワーク・ロードのモデルとして，実際のジョブ実行のトレース・データを利用したり，サンプリングした実測データを入力するようなタイプのシミュレータがある。この型に属するものは，イベントごとのトレース・データを利用するAMAP，サンプリング・データを利用するものとしてCUSIMがある。これらのタイプのものは，実測データを収集し，解析するという過程ではソフトウェア・モニタと同じ機能を果しているわけだが，一度収集したワーク・ロード・データに，種々のシステム構成を対応させて，シミュレートすることが第一目的と考え，シミュレータの範ちゅうに入れている。

次項には，これら3つのタイプのシミュレータの代表例としてCASE，CSS，AMAP をとりあげ，その概要を説明している。

3.5.2 専用シミュレータの実例

A. *CASE

(1) 概　要

CASE（Computer-Aided System Evaluation）は，米国のTesdata Systems Corpで開発された商用のコンピュータ・システム・シミュレータである。

CASEを利用するユーザは，システム構成，負荷を入力として与えることにより，そのシステムのパフォーマンスに関する各種のデータを出力レポートとして得ることができるようになっている。

その利用分野は，システム設計，システム構成の決定，装置の選択，システムの検査等，広範囲に渡っている。

＊CASEについてはTesdata社より提供を受けた"CASE general information manul"より図表などを引用させて頂いた。

CASEシステムは，CASEライブラリとCASEプログラムから構成されているが，CASEプログラムはFORTRANで記述されているので，比較的マシン・インデペンデントな性格を備えていると言える。

現在IBM 360/370 OS，CDC 6000/7000 SCOPE，UNIVAC 1100 EXECⅡ・EXECⅧ，HIS 600/6000 GECOS などの下で，利用が可能であり，米国内および欧州，日本で利用されている。

CESE によるシミュレーションには以下の5つの要素が含まれ，それらの関係は図 3−37 のようになる。

① 評価するシステムの定義

② CASE プログラム

③ CASE ライブラリ

④ システムの最適化

⑤ レポート出力

以下，これらの各要素について簡単に説明する。

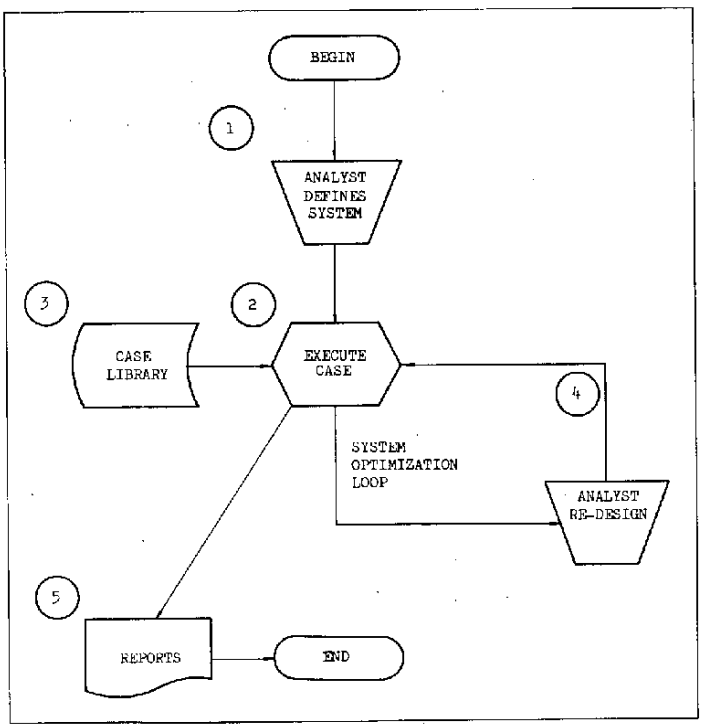


図3－37　CASE SYSTEM FLOW

(2) システムの定義

CASE を用いて，あるコンピュータ・システムを評価しようとした場合，入力データとしての基本的なものは，ワーク・ロードとコンフィギュレーションに関するデータである。

ここで定義するワーク・ロードは，知られている限り詳細に記述してかまわないが，初期の段階では，ファイルに関する一般的な特性などを記述しておき，シミュレーションの過程が繰り返されて進むにつれて，より多くの情報が得られるので，より正確なシミュレーションをおこなうためのデータはこの過程で除々に定義していくことになる。

初期段階で定義するワーク・ロード・データとしては以下のようなものがある。

① 利用されるファイルの定義

ファイル名称，レコード総数，レコード長，ブロッキング・ファクタ，デバイス・タイプ，編成，レコードあたりり英・数字のフィールド数などである。

② 実行に関する定義

処理プロセスの定義はマクロ，サブルーティン，命令を用いた具体的な形でも定義出来るし，編集したり修正したりするフィールドの数を定義する形式もある。又プロセスの使用頻度は確率分布の形で与えることが出来る。

ることが出来る。

③ 実行順序に関する定義

JOB 間の因果関係を定義する。

スケジューリングの情報を与える。

また，コンフィギュレーションの定義として入力するデータは以下のようなものである。

・ハードウェア，ソフトウェアの構成

例えば，メーカー名，モデル番号，記憶容量，チャネルの数とタイプ，

カードリーダ，プリンター，磁気テープ，ダイレクト・アクセス・ディバイス，各要素間の接続関係等。

これらの入力データは，定形のコーディング・シートが用意されており，必要なシートに必要な項目を記入すれば良いようになっている。

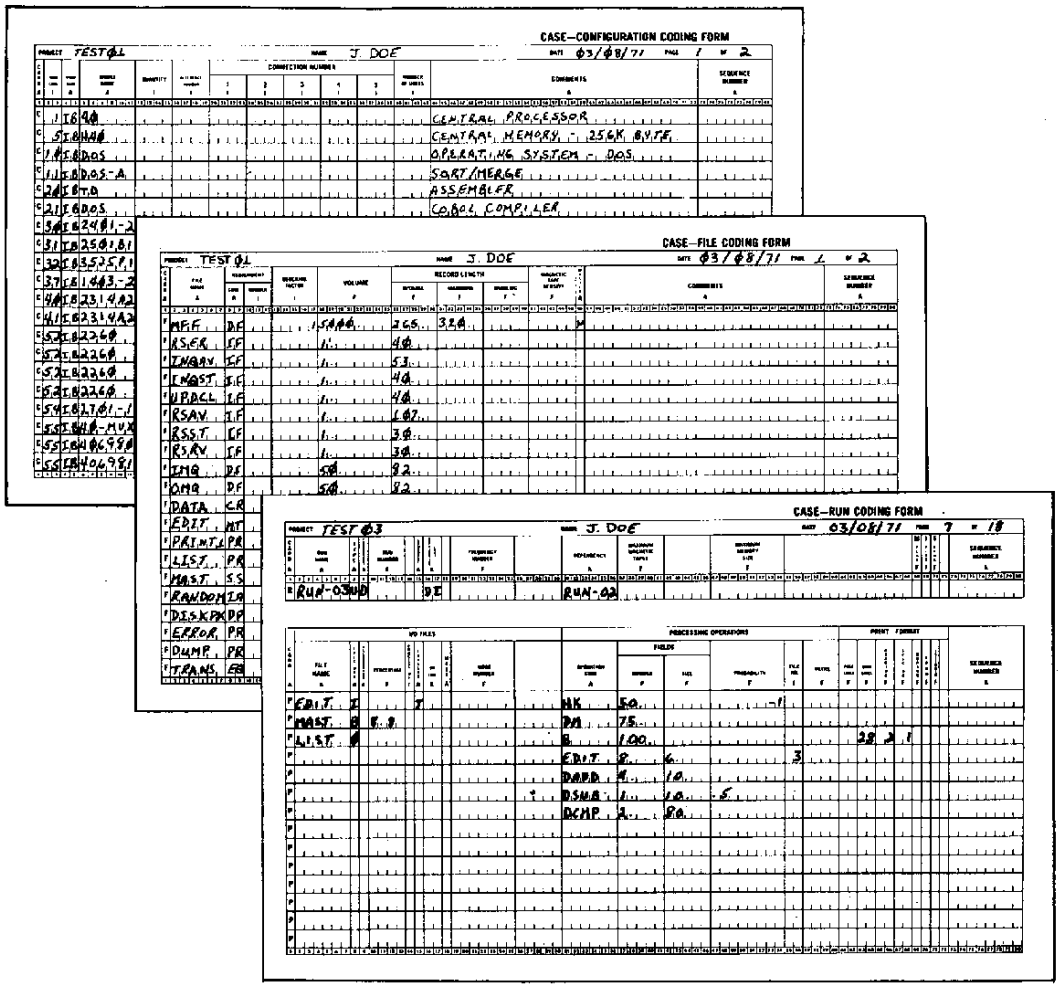


図3－38　CASE入力データ・コーディング・シート

また，これらの入力データは，前に述べたアカウンティング・データやハードウェア・モニタによるレポート，予算的な制約，プログラム説明書などを源として各利用者が記入することになる。

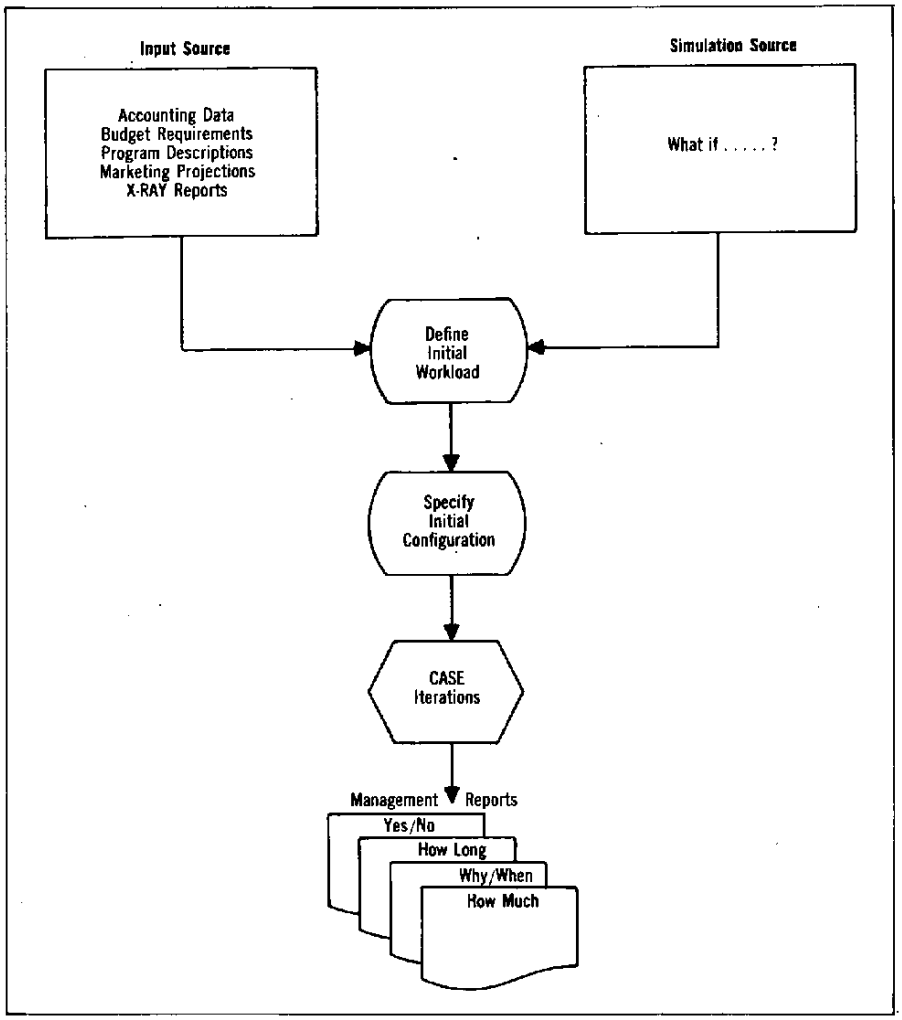


図3－39　CASEへの入力

(3)　CASEプログラム

CASE プログラムは，次の2つから成っている。

IPA（Independent Processing Analyzer）

CPA（Concurrent Processing Analyzer）

①　IPA

IPAは，プログラムが単独で実行される（Batch mode）時の状態をシミュレートするもので，ユーザの定義したコンフィギュレーショ

ン，ワーク・ロードを入力としてハードウェア・ソフトウェアの特性を考慮してシミュレートし，システムの利用状況に関する情報を出力するものである。ＩＰＡにより出力される情報としては，スペースの要求量，内部処理時間，ファイル入出力時間，全実行時間などがある。

ＩＰＡの出力は，そのままＣＰＡの入力として利用されるが，ＩＰＡだけの単独実行も可能である。

② ＣＰＡ

ＣＰＡはＩＰＡからの結果に加えて，ユーザが指定したスケジューリング情報とリアル・タイム制御情報をもとに，ランのスケジュールを決定し，マルチ・プログラミング・モード（多重処理）あるいはリアル・タイム・モードでの処理状態におけるワーク・ロードをシミュレートするものである。ＣＰＡによるシミュレーションでは，各資材の利用統計の他に各ランの優先関係，ある時間間隔におけるあるモジュールの活動状況，各ランの開始・終了時刻などの状態を示すレポートが得られる。

ＣＰＡを利用し，あるシステム構成のもとで，各ランがスケジューリングされる過程をシミュレートすることにより，システム・アナリストは，最適のシステム・コンフィギュレーション，システム設計に等々有用な情報を得ることができる。

(4) ＣＡＳＥライブラリ

各メーカのハードウェア，ソフトウェアの特性データを蓄えているのがＣＡＳＥライブラリで，ＣＡＳＥによるシミュレーションの入力の一つとして使用される。

シミュレータは，ライブラリから評価対象のハードウェア・ソフトウェアに関する技術的な特性を抽出して，ユーザの入力データとともにモデルをつくりあげ，シミュレートすることになる。

ライブラリに蓄えられている特性データの内容は，次のようなものである。

ハードウェア：本体，周辺装置，通信装置等の特性

ソフトウェア：OS，アセンブラ，各種言語，各種ユーティリティ，ソート，RPGなどのテクニカル・データ

これらのライブラリ中のデータは常に最新の状態にアップデートされているが，現在，含まれている機種は，IBM，CDC，UNIVAC，Burroughs，HIS，NCR，DEC，Philco，XDS などで，近々ICL—1900シリーズに関するデータも追加されるとのことである。

(5) システムの最適化

CASEシミュレーションにおける，重要な要素は，"Man in the loop"と呼ばれるシステム・アナリストである。アナリストは通常，コンフィギュレーションの初期構成を用意し，要求される最終解に近ずくようコンフィギュレーションや設計を適合（アジャスト）させていくような繰返しを行うため，上記のように呼ばれるわけである。

このようなアジャストメントは，IPA，CPAからの各種のレポートを解析することによっておこなわれ，解析・アジャストを繰り返すことにより，望む結果に除々に近づけていくわけであるが，これらの繰り返しの過程を簡単化・迅速化するために，CASEではコントロール・カードの変更だけで済むように配慮されている。アジャストメントには，設計上のアジャストメントと，機器構成上のアジャストメントの2つがある。

設計上のアジャストメントは，何回かの準備的なシミュレーションの後におこなわれるもので，CASEでは，システム内容を単純化したり決定するためにファイルや実行に関するレポートを示してくれるようになっている。

機器構成上のアジャストメントは設計上の変更と組みあわせて行うか，単独におこなうが，これには以下のようなものが考えられる。

- チャネル，コントローラ，メモリなどの追加削除
- マス・ストレージの種類・量の変更
- 磁気テープ，カード・リーダ，パンチ，LPなどの台数・パフォーマンスの変更

このようなアジャストを繰り返すことによりアナリストは，システムのパフォーマンスを向上させるための重要な要素として，働くわけである。

このような繰返しのプロセスを図 3−40 に示す。

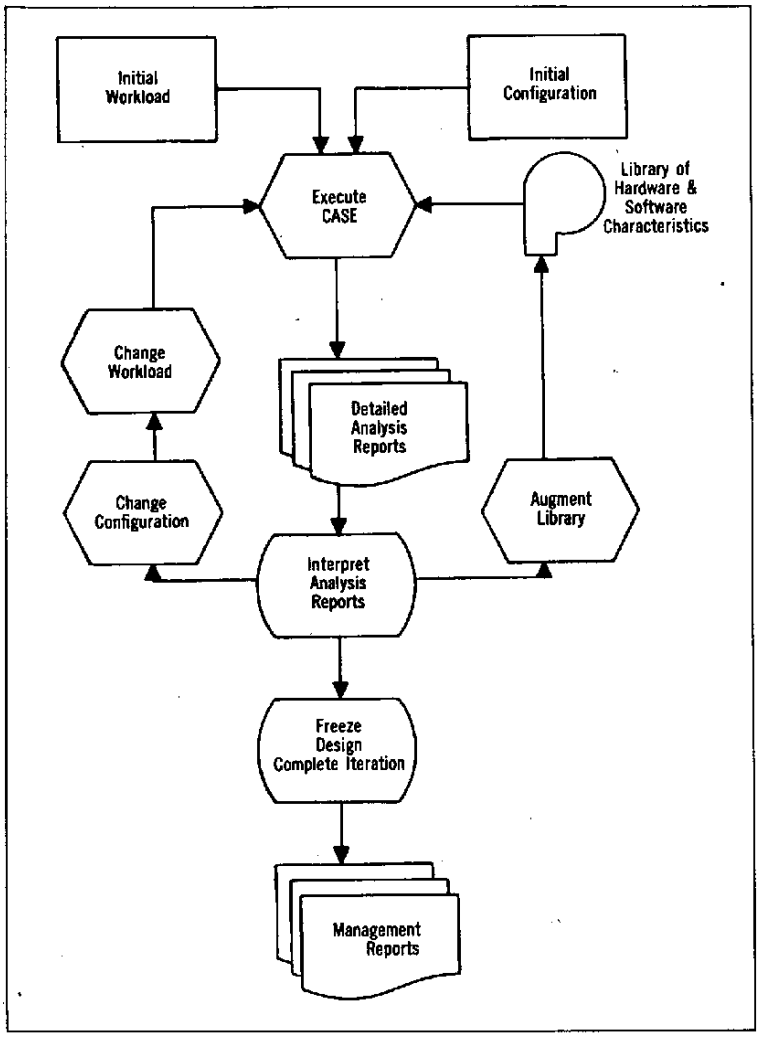


図3−40　CASEによるシミュレーションの過程

(6)　CASEの出力するレポート

CASE の出力するレポートには 2つのタイプがある。1つはアナリシス・レポートであり，もう1つはマネージメント・レポートである。

両方とも各シミュレーション・ランの結果出力されるが，前者は，その結果を見て次のステップ（シミュレーション）に進むのに都合の良いように，解析者への便宜をはかっている。後者は，各シミュレーションの繰り返しの終了時に出力するレポートで，指定によりかなり詳細な内容も出力でき，最終ドキュメントとして利用される。

CASEの作り出すレポートは種類にしておよそ45種類準備されている。これ等は，ユーザの指示に従って必要なものだけが作成される。

〔IPAの出力するレポート〕

Configuration Report
File Description Report
Run Description Report
Run Detail Report
Run Summary Report
Unit Utilization Report
Channel Utilization Report
File Summary Report
Summary Report

〔CPAの出力するレポート〕

Job Scheduling Constraint Report
Batch Job Analysis Report
Multiprogrammed Central Site Response Report
Concurrent Processing Slice Report
Configuration Macroqueing Report
Communications Network Queuing Report
Real-Time Activity Report

図3-41から図3-43は，IPAから出力されたレポートではあるが，利用者が入力したコンフィギュレーション，ワークロード・データを整理・編集したものである。

CONFIGURATION REPORT FOR IBM 360/40 03/08/71

ITEM NR.	NAME	MANF.	MODEL	QUANTITY	REFERENCE NR.	CONNECTIONS 1	2	3	4	5
1	CENTRAL PROCESSOR	IBM	40	1	-0	-0	-0	-0	-0	-0
2	CENTRAL MEMORY	IBM	M40	1	-0	-0	-0	-0	-0	-0
3	OPERATING SYSTEM	IBM	DOS	1	-0	-0	-0	-0	-0	-0
4	SORT	IBM	DOS-A	1	-0	-0	-0	-0	-0	-0
5	ASSEMBLER	IBM	TO	1	-0	-0	-0	-0	-0	-0
6	COBOL COMPILER	IBM	DOS	1	-0	-0	-0	-0	-0	-0
7	MAGNETIC TAPE	IBM	2401-2	6	1	4	-0	-0	-0	-0
8	CARD READER	IBM	2501B1	1	1	3	-0	-0	-0	-0
9	CARD PUNCH	IBM	3525P1	1	1	3	-0	-0	-0	-0
10	PRINTER	IBM	1403-2	1	1	5	-0	-0	-0	-0
11	DISK FILE	IBM	231442	2	1	2	-0	-0	-0	-0
12	DISK PACK	IBM	231442	2	1	2	-0	-0	-0	-0
13	TERMINAL	IBM	2260	1	21	29	-0	-0	-0	-0
14	TERMINAL	IBM	2260	1	22	29	-0	-0	-0	-0
15	TERMINAL	IBM	2260	1	23	30	-0	-0	-0	-0
16	TERMINAL	IBM	2260	1	24	30	-0	-0	-0	-0
17	INTERFACE	IBM	2701-1	1	6	3	-0	-0	-0	-0
18	CHANNEL	IBM	40-MUX	1	3	-0	-0	-0	-0	-0
19	CHANNEL	IBM	406980	1	1	-0	-0	-0	-0	-0
20	CHANNEL	IBM	406981	1	2	-0	-0	-0	-0	-0
21	CONTROLLER	IBM	2803-1	1	4	1	-0	-0	-0	-0
22	CONTROLLER	IBM	2821-2	1	5	3	-0	-0	-0	-0
23	COMMUNICATION COMP	IBM	2701-2	1	32	6	-0	-0	-0	-0
24	COMMUNICATION COMP	IBM	2701-2	1	33	6	-0	-0	-0	-0
25	COMMUNICATION COMP	IBM	2848-2	1	29	32	-0	-0	-0	-0
26	COMMUNICATION COMP	IBM	2848-2	1	30	33	-0	-0	-0	-0

図 3－41

FILE DESCRIPTION REPORT PAGE NR. 1

NR.	NAME	RECORDS	RECORD SIZE (CHAR) AVG	MAX	NUMERICS	BLOCK FACTOR	ASSIGNMENT	DENSITY (FPI)	TYPE	TOTAL SIZE (CHAR)
1	MFF	5000	265	320	-0	1	DF -0	-0	M	1325000
2	RSER	1	40	40	-0	1	IF -0	-0		40
3	INQAV	1	53	53	-0	1	IF -0	-0		53
4	INQST	1	40	40	-0	1	IF -0	-0		40
5	UPDCL	1	40	40	-0	1	IF -0	-0		40
6	RSAV	1	107	107	-0	1	IF -0	-0		107
7	RSST	1	30	30	-0	1	IF -0	-0		30
8	RSRV	1	30	30	-0	1	IF -0	-0		30
9	INQ	50	82	82	-0	1	DF -0	-0		4100
10	OMQ	50	82	82	-0	1	DF -0	-0		4100
11	DATA	10000	80	80	-0	-0	CR -0	-0		800000
12	EDIT	9000	80	80	20	20	MT 4	800		720000
13	PRINT1	1000	120	120	-0	-0	PR -0	-0		120000
14	LIST	9000	120	120	-0	-0	PR -0	-0		1080000
15	MAST	80000	200	250	50	-0	SS -0	-0	M	16000000
16	RANDOM	42000	135	135	40	-0	IA -0	-0	M	5670000
17	DISKPK	8700	80	80	20	-0	DP -0	-0		696000
18	ERROR	300	120	120	-0	-0	PR -0	-0		36000
19	DUMP	8700	120	120	-0	-0	PR -0	-0		1044000
20	TRANS	15000	80	80	-0	-0	EO 1	-0		1200000
21	SPOOL	15000	80	80	-0	-0	SS -0	-0		1200000
22	PUNCH	5500	80	80	-0	-0	CP -0	-0		440000
23	UPTRAN	4000	80	80	-0	-0	EO 1	-0		320000
24	INDEX	50000	400	400	30	-0	DF -0	-0	M	20000000
25	PRINT2	4000	120	120	-0	-0	PR -0	-0		480000
26	EXTRAK	7000	175	175	30	-0	RP -0	-0		1225000
27	SORTED	7000	175	175	30	-0	SS -0	-0		1225000
28	CARD	1500	80	80	-0	-0	CR -0	-0		120000
29	COBOLC	200	80	80	15	-0	CR -0	-0		16000
30	TESTX	20000	2000	2000	-0	-0	RP 110	-0	M	40000000
31	PRINT3	7000	120	120	-0	-0	PR -0	-0		840000
32	UTILF	200	45	45	10	-0	DF 1	-0		9000
33	COBLST	220	132	132	-0	-0	PR -0	-0		29040
34	LOADC	200	80	80	-0	-0	SS -0	-0		16000
35	UTIL	200	45	45	10	-0	DF 1	-0		9000

図 3－42

RUN DESCRIPTION REPORT FOR RUN = RUN-03 (DI) NR.= 14

FREQUENCY = 22.000 TIMES PER MONTH RUN TYPE = UPDATE

PROGRAMMING LANGUAGE = COBOL MUST FOLLOW RUN = RUN-02

RUN FILES

IO	NAME	NR.	RECORDS	PERCENT USED	RECORD SIZE	BLOCK FACTOR	PROCESS-ING	ALLOCA-TION	OFF-LINE	MODE	PREVIOUS RUN	ASSIGN-MENT	DENSITY	TRAN
I	EDIT	12	9000	100	80	20			0	0	13	MT 4	800	T
B	MAST	15	80000	9	200	10			0	0	0	MT 0	-0	
O	LIST	14	9000	100	120	-0		A	1	0	0	14 0	-0	

PRINT FORMAT

PAGE SIZE	DATA LINES	HEADING LINE	SPACING	COPIES	SPECIAL FORM	CHARACTER SET	FILE
66	28	2	1	-0			LIST

PROCESSING ACTIVITY

OPERATION	PROBABILITY	FILE	NUMBER	LENGTH	PATHS
MK	-1.00	T FILE	50	-0	1
DM	1.00	T FILE	75	-0	1
B	1.00	T FILE	100	-0	1
EDIT	1.00	LIST	8	6	1

図3－43

図3-44は，単独に実行されたプログラムをシミュレートし解析した結果をＩＰＡが出力したものである。

この結果は，プログラム設計の際にも非常に有用な情報となる。

RUN DETAIL REPORT FOR RUN = RUN-03 (DI) NR.= 14

FREQUENCY = 22.000 TIMES PER MONTH RUN TYPE = UPDATE

RUN FILES

IO	NAME	NR.	RECORDS USED	RECORD SIZE	BLOCK FACTOR	ASSIGNMENT MEDIA		ASSIGNMENT MODEL	REELS OR PACKS	I/O CHANNEL	RUN DELAY TIME	I/O UNIT TIME
I	EDIT	12	9000	70	20	MT	4	2401-2	1	1	.173	.273
I	MAST	15	7040	175	10	MT	2	2401-2	1	1	3.575	5.634
O	LIST	14	9000	120	29	DF	1	2314A2	1	2	0.000	.132
O	MAST	15	7040	175	10	MT	1	2401-2	1	1	3.575	5.634

MEMORY REQUIREMENTS		PROCESSOR REQUIREMENTS		OVERHEAD REQUIREMENTS		TOTAL TIMES	
PROGRAM	3009	PROGRAM	3.016	INITIALIZATION	.001	DELAY	.284
I/O	22936	I/O	.670	I/O SET-UP	1.000	EXECUTION	11.824
SYSTEM	12488	SYSTEM	.530	I/O CHANGE	0.000	CHARGE	12.888
TOTAL	38433	TOTAL	4.217	I/O COMPLETION	1.062	ELAPSED	13.888

UNITS, SIZES IN ALPHA CHARACTERS. TIME IN MINUTES.

図3－44

図3-45は，図3-44のRUN Detail report をサマライズしたもので，異常に処理時間の長いものや，メモリや装置の占有が多いものにつ

いては，特別なしるしをつけて注意を引くように考慮されている。

RUN SUMMARY REPORT (DAILY)　　PAGE NO. 1

NO.	NAME	APPLI-CATION	FREQUENCY	TYPE	MEMORY (CHAR)	MT	UNITS OF DF	OP	FILES CR	PR	USED CP	XX	TIMES (MIN.) PROCESSOR	CHARGE	ELAPSED
1	ICC1		88.000		1534	0	1	0	0	0	0	1	.678	.725	.725
2	ICC2		50.000		209	0	1	0	0	0	0	1	.395	.412	.412
3	ICC3		68.000		200	0	1	0	0	0	0	1	.524	.560	.560
4	PAVAIL		61.000		1740	0	0	0	0	0	0	0	.016	.016	.016
5	PSTAT		35.000		1560	0	0	0	0	0	0	0	.005	.005	.005
6	RESERV		47.000		476	0	0	0	0	0	0	0	.001	.001	.001
7	OCC1		66.000		1815	0	1	0	0	0	0	1	.508	.544	.544
8	OCC2		37.000		200	0	1	0	0	0	0	1	.285	.305	.305
9	OCC3		51.000		200	0	1	0	0	0	0	1	.393	.420	.420
10	PERROR		33.000		900	0	0	0	0	0	0	0	.003	.003	.003
11	CANCEL		17.000		1650	0	0	0	0	0	0	0	.002	.002	.002
12	RUN-01	DI	1.000		9767	1	0	0	1	1	0	0	3.437	18.532	19.532
13	RUN-02	DI	1.000	SORT	64200	1	0	1	0	0	0	0	.603	2.727	4.727
14	RUN-03	DI	1.000	UPDATE	26145	3	1	0	0	0	0	0	4.217	12.888	13.888
-14	RUN-03A	DI	1.000	XFER	12422	0	1	0	0	1	0	0	1.296	19.380	19.380
15	RUN-04	DI	1.000		27308	1	0	1	0	1	0	0	3.963	4.411	5.411
16	RUN-05	DI	1.000	REPORT	16200	0	0	1	0	1	0	0	1.320	19.063	20.563
17	RUN-06	DI	1.000	XFER	15440	0	1	0	1	0	0	0	2.224	27.557	27.557
18	RUN-07	DI	1.000		26754	0	2	1	0	0	0	0	3.229	3.238	4.238
-18	RUN-07N	DI	1.000	XFER	12400	0	1	0	0	0	1	0	.775	56.490	56.490
19	RUN-08	SA	2.000	UPDATE	23265	0	2	0	1	0	0	0	5.525	16.277	16.277
-19	RUN-08A	SA	2.000	XFER	12422	0	1	0	0	1	0	0	1.159	13.853	14.853
20	RUN-09	SA	2.000	EXTRACT	21021	0	1	1	0	0	0	0	19.632	19.635	21.635
21	RUN-10	SA	2.000	SORT	100200	4	1	1	0	0	0	0	2.153	8.046	14.046
22	RUN-11	SA	2.000	EXTRACT	23540	0	2	0	1	0	0	0	1.660	5.515	5.515
-22	RUN-11A	SA	2.000	XFER	12422	0	1	0	0	1	0	0	2.016	23.500	23.500
23	RUN-12	CO	4.000	XFER	11426	0	1	0	1	0	0	0	.142	1.477	1.477
24	RUN-13	CO	4.000	TRANSLATE	24680	0	5	0	0	0	0	0	1.926	2.344	2.344
-24	RUN-13A	CO	4.000	XFER	12326	0	1	0	0	1	0	0	.157	1.659	1.659

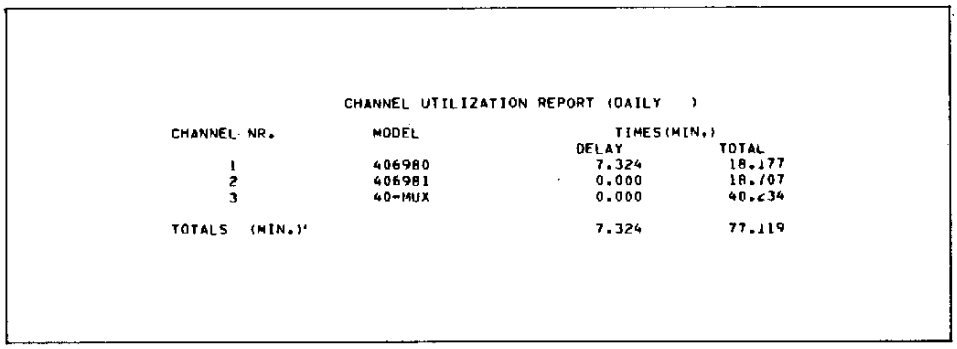
図3－45

図3－46から図3－48は，各装置，チャネル，ファイルごとにすべてのIPA シミュレーションが終了したあと，サマリとして出力するレポートである。

UNIT UTILIZATION REPORT (DAILY)

UNIT	ASSIGNMENT CODE		MODEL	CHANNEL (FIRST)	TIMES (MIN.) DELAY	TOTAL	IA ALLOCATION (M.CHAR.) MASTER	PERMANENT	TOTAL
MAGNETIC TAPE	MT	1	2401-2	1	0.000	11.177	0.000	0.000	0.000
MAGNETIC TAPE	MT	2	2401-2	1	0.000	5.634	0.000	0.000	0.000
MAGNETIC TAPE	MT	3	2401-2	1	0.000	0.000	0.000	0.000	0.000
MAGNETIC TAPE	MT	4	2401-2	1	0.000	1.386	0.000	0.000	0.000
MAGNETIC TAPE	MT	5	2401-2	1	0.000	0.000	0.000	0.000	0.000
MAGNETIC TAPE	MT	6	2401-2	1	0.000	0.000	0.000	0.000	0.000
CARD READER	CR	1	2501R1	3	45.442	67.602	0.000	0.000	0.000
CARD PUNCH	CP	1	3525P1	3	55.712	56.688	0.000	0.000	0.000
PRINTER	PR	1	1403-2	3	71.492	88.063	0.000	0.000	0.000
DISK FILE	DF	1	2314A2	2	0.000	2.935	10.000	1.488	16.353
DISK FILE	DF	2	2314A2	2	10.748	26.055	25.770	0.000	25.770
DISK PACK	DP	1	2314A2	2	0.000	5.636	0.000	0.000	146.792
DISK PACK	DP	2	2314A2	2	0.000	0.000	0.000	0.000	0.000
INTERFACE	IF	6	2701-1	3	0.000	.637	0.000	0.000	0.000
TOTALS (MIN.)					183.394	257.593	35.770	1.488	188.915

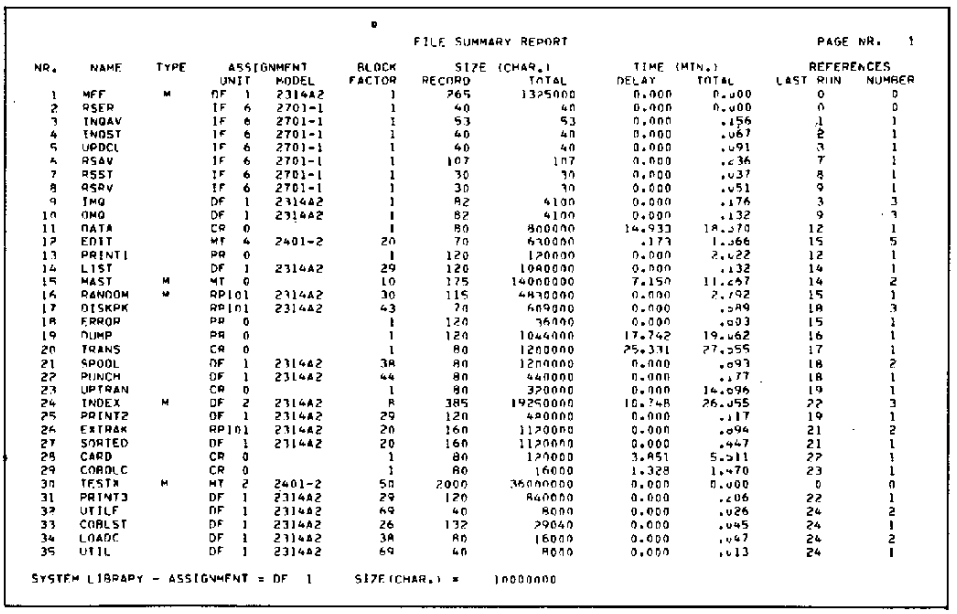
図3－46　ユニット利用状況



CHANNEL UTILIZATION REPORT (DAILY)

CHANNEL NR.	MODEL	TIMES(MIN.) DELAY	TOTAL
1	406980	7.324	18.177
2	406981	0.000	18.707
3	40-MUX	0.000	40.234
TOTALS (MIN.)		7.324	77.119

図３－４７　チャネル利用状況



FILE SUMMARY REPORT　　PAGE NR. 1

NR.	NAME	TYPE	ASSIGNMENT UNIT	MODEL	BLOCK FACTOR	SIZE (CHAR.) RECORD	TOTAL	TIME (MIN.) DELAY	TOTAL	REFERENCES LAST RUN	NUMBER
1	MFF	M	DF 1	2314A2	1	765	1325000	0.000	0.000	0	0
2	RSER		IF 6	2701-1	1	40	40	0.000	0.000	0	0
3	INQAV		IF 6	2701-1	1	53	53	0.000	.156	1	1
4	INQST		IF 6	2701-1	1	40	40	0.000	.067	2	1
5	UPDCL		IF 6	2701-1	1	40	40	0.000	.091	3	1
6	RSAV		IF 6	2701-1	1	107	107	0.000	.236	7	1
7	RSST		IF 6	2701-1	1	30	30	0.000	.037	8	1
8	RSRV		IF 6	2701-1	1	30	30	0.000	.051	9	1
9	IMQ		DF 1	2314A2	1	82	4100	0.000	.176	3	3
10	OMQ		DF 1	2314A2	1	82	4100	0.000	.132	9	3
11	DATA		CR 0		1	80	800000	14.933	18.370	12	1
12	EDIT		MT 4	2401-2	20	70	630000	.173	1.066	15	5
13	PRINT1		PR 0		1	120	120000	0.000	2.622	12	1
14	LIST		DF 1	2314A2	29	120	1040000	0.000	.132	14	1
15	MAST	M	MT 0		10	175	14000000	7.150	11.267	14	2
16	RANDOM	M	RP101	2314A2	30	115	4830000	0.000	2.792	15	1
17	DISKPK		RP101	2314A2	43	70	609000	0.000	.589	18	3
18	ERROR		PR 0		1	120	36000	0.000	.003	15	1
19	DUMP		PR 0		1	120	1044000	17.742	19.062	16	1
20	TRANS		CR 0		1	80	1280000	25.331	27.555	17	1
21	SPOOL		DF 1	2314A2	38	80	1280000	0.000	.093	18	2
22	PUNCH		DF 1	2314A2	44	80	440000	0.000	.177	18	1
23	UPTRAN		CR 0		1	80	320000	0.000	14.096	19	1
24	INDEX	M	DF 2	2314A2	R	385	19250000	16.748	26.055	22	3
25	PRINT2		DF 1	2314A2	29	120	480000	0.000	.117	19	1
26	EXTRAK		RP101	2314A2	20	160	1120000	0.000	.094	21	2
27	SORTED		DF 1	2314A2	20	160	1120000	0.000	.447	21	1
28	CARD		CR 0		1	80	120000	3.851	5.511	22	1
29	COBOLC		CR 0		1	80	16000	1.328	1.470	23	1
30	TESTM	M	MT 2	2401-2	50	2000	36000000	0.000	0.000	0	0
31	PRINT3		DF 1	2314A2	29	120	840000	0.000	.206	22	1
32	UTILF		DF 1	2314A2	69	40	8000	0.000	.026	24	2
33	COBLST		DF 1	2314A2	26	132	29040	0.000	.045	24	1
34	LOADC		DF 1	2314A2	38	80	16000	0.000	.047	24	2
35	UTIL		DF 1	2314A2	69	40	8000	0.000	.013	24	1

SYSTEM LIBRARY - ASSIGNMENT = DF 1　　SIZE(CHAR.) = 10000000

図３－４８　ファイル利用状況

図３－４９は，CPA 実行時に出力されるレポートであるが，多重処理における，各ジョブの前後関係を利用者が入力したスケジューリング情報のサマリである。

CASE JOB SCHEDULING CONSTRAINT REPORT PAGE NO. 1

JOB NAME	JOB NUMBER	PREDECESSOR JOBS 1	2	3	4	5	SUCCESSOR JOBS 1	2	3	4	5
RUN-13A	1	RUN-13									
RUN-13	2	RUN-12					RUN-13A				
RUN-12	3						RUN-13				
RUN-11A	4	RUN-11									
RUN-11	5						RUN-11A				
RUN-10	6	RUN-09									
RUN-09	7	RUN-08					RUN-10				
RUN-01	8						RUN-02				
RUN-02	9	RUN-01					RUN-04	RUN-03			
RUN-03	10	RUN-02					RUN-03A				
RUN-03A	11	RUN-03									
RUN-04	12	RUN-02					RUN-06	RUN-05			
RUN-05	13	RUN-04									
RUN-06	14	RUN-04					RUN-07				
RUN-07	15	RUN-06					RUN-07N				
RUN-07N	16	RUN-07									
RUN-08	17						RUN-08A	RUN-09			
RUN-08A	18	RUN-08									

図３－４９　スケジューリング情報

以下，図３－５０から図３－５６まで CPAの出力するレポートを参考までにのせておく。

MULTIPROGRAMMED CENTRAL SITE RESPONSE TIMES FOR RT JOBS

INTERVAL FROM 1:09:30 TO 1:09:36 (DAYS:HRS:MINS)

JOB NAME	JOB NUMBER	NO. OF EXECUTIONS	EXECUTION TIME	WAIT TIME	RESPONSE TIME (MIN:SEC) AVERAGE	MAXIMUM
WZ901	1	24	0:00.081	0:00.000	0:00.083	0:00.132
WZ201	2	5	0:00.431	0:03.000	0:04.275	0:05.178
WZ001	3	44	0:00.251	0:00.000	0:00.252	0:00.523
WY101	4	32	0:00.106	0:02.880	0:02.990	0:03.609
WY001	5	32	0:00.108	0:00.000	0:00.126	0:00.844
TMX01	6	21	0:00.786	0:03.120	0:04.868	0:07.601
EE001	7	10	0:00.655	0:03.059	0:04.867	0:07.263
AF001	8	298	0:00.470	0:03.000	0:04.580	0:12.201
AB001	9	347	0:00.477	0:00.000	0:01.035	0:04.262
BE001	10	6	0:01.932	0:00.000	0:04.981	0:14.259
BA001	11	38	0:02.237	0:03.000	0:07.214	0:12.880
CA101	12	7	0:01.958	0:03.059	0:07.755	0:13.633
AG001	13	17	0:01.219	0:00.000	0:04.564	0:11.487
AF101	14	14	0:01.860	0:03.120	0:07.451	0:16.557
AA101	15	27	0:01.874	0:03.120	0:06.488	0:13.118

CASE

BATCH JOB REPORT

INTERVAL FROM 1:09:00 TO 1:17:06 (DAYS:HRS:MINS)

JOB NAME	JOB NUMBER	AVAILABLE TIME (HRS:MIN:SEC)	INITIATION TIME (HRS:MIN:SEC)	COMPLETION TIME (HRS:MIN:SEC)	TOTAL TIME (HRS:MIN:SEC)	M.P. EFFICIENCY
RUN-01	12	9:00:00	9:00:00	9:19:31	19:31.933	1.00
RUN-02	13	9:19:31	9:19:31	9:24:15	4:43.640	1.00
RUN-0A	21	9:00:00	9:19:31	9:27:40	8:08.303	1.00
RUN-11	9	9:00:00	9:27:40	9:30:25	2:45.447	1.00
RUN-0AA	22	9:27:40	9:27:40	9:36:49	9:09.260	.81
RUN-03	14	9:24:15	9:24:15	9:41:01	16:46.139	.83
RUN-12	7	9:00:00	9:41:01	9:41:27	0:25.644	.86
RUN-13	6	9:41:27	9:41:27	9:42:35	1:07.821	.52
RUN-09	11	9:27:40	9:30:25	9:44:44	14:18.405	.76
RUN-11A	8	9:30:25	9:36:49	9:50:20	13:30.714	.87
RUN-10	10	9:44:44	9:44:44	9:51:45	7:01.389	1.00
RUN-04	16	9:24:15	9:50:20	9:55:44	5:24.684	1.00
RUN-03A	15	9:41:01	9:55:44	10:15:07	19:22.784	1.00
RUN-06	18	9:55:44	9:55:44	10:23:18	27:33.406	1.00
RUN-07	19	10:23:18	10:23:18	10:27:32	4:13.796	1.00
RUN-05	17	9:55:44	10:15:07	10:35:41	20:33.758	1.00
RUN-12	7	10:45:00	10:45:00	10:45:22	0:22.150	1.00
RUN-13	6	10:45:00	10:45:00	10:45:35	0:35.162	1.00
RUN-07N	20	10:27:32	10:27:32	11:24:01	56:29.371	1.00
RUN-13A	5	10:45:00	11:24:01	11:24:26	0:24.884	1.00
RUN-13A	5	9:42:35	11:24:26	11:24:51	0:24.884	1.00
RUN-12	7	12:30:00	12:30:00	12:30:25	0:25.717	.86
RUN-13A	5	12:30:00	12:30:00	12:30:31	0:31.434	.79
RUN-13	6	12:30:00	12:30:00	12:30:52	0:52.829	.67
RUN-08	21	12:30:00	12:30:52	12:43:19	12:26.345	.65
RUN-09	11	12:30:00	12:30:25	12:46:33	16:07.719	.67
RUN-11A	8	12:30:00	12:30:31	12:47:32	17:00.702	.69
RUN-11	9	12:30:00	12:46:33	12:49:18	2:45.447	1.00
RUN-10	10	12:30:00	12:43:19	12:51:07	7:47.919	.90
RUN-08A	22	12:30:00	12:47:32	12:54:57	7:25.602	1.00
RUN-12	7	14:15:00	14:15:00	14:15:23	0:23.829	.93
RUN-13A	5	14:15:00	14:15:00	14:15:26	0:26.563	.94
RUN-13	6	14:15:00	14:15:00	14:15:36	0:36.841	.95

図3－51

CASE

CONCURRENT PROCESSING SLICE REPORT

PAGE NR. 2

NUMBER		TIME (HRS:MIN:SEC)	JOBS STARTED	JOBS COMPLETED	RESOURCE PROFILE QUANTITY	QUEUES
SLICE	4 ON	2:00:09.238	0105 (61)	M .67	MT 1	
ACTIVE	5 SLICE	:53.019		MT 1		
JOB QUEUE	1 OFF	2:01:02.257		MT 7		
				MT 7		
				CR 1		
				PR 2		
				DF .57		

SLICE MICROQUEUING ANALYSIS

JOB	PRIORITY	PROGRESS RATE	PRINCIPLE CAUSE OF DELAY
K011 (17)	1	1.00	
N031 (3)	1	1.00	
D105 (61)	1	1.00	
U031 (14)	1	.88	DF 12
U012 (12)	1	1.00	

SLICE EFFECTIVE PROGRESS RATE - 4.88

NUMBER		TIME (HRS:MIN:SEC)	JOBS STARTED	JOBS COMPLETED	RESOURCE PROFILE QUANTITY	QUEUES
SLICE	5 ON	2:01:02.257	N111 (9)	N111 (9)	M .68	
ACTIVE	5 SLICE	1:53.497			MT 1	
JOB QUEUE	0 OFF	2:02:55.755			MT 7	
					MT 7	
					CR 1	
					PR 1	
					DF .57	

SLICE MICROQUEUING ANALYSIS

JOB	PRIORITY	PROGRESS RATE	PRINCIPLE CAUSE OF DELAY
K011 (17)	1	1.00	
N031 (3)	1	1.00	
N111 (9)	1	1.00	
U031 (14)	1	.88	DF 12
U012 (12)	1	1.00	

SLICE EFFECTIVE PROGRESS RATE - 4.88

図3－52

CASE — CONFIGURATION MACROQUEUING REPORT — PAGE NR. 1

INTERVAL FROM 1:00:00 TO 1:17:00 (DAYS:HRS:MINS)

GROUP NUMBER	GROUP CODE	GROUP MODULE NAME	TOTAL COMPONENTS	COMPONENTS IN USE AVG	COMPONENTS IN USE MAX	PERCENTAGE OF GROUP USED	QUEUE LENGTH AVG	QUEUE LENGTH MAX	QUEUE LENGTH DEV	TOTAL DELAY (HRS:MINS:SECS)
2	M	CENTRAL MEMORY	1	.11	.58	11.3	0.00	0	0.00	:00.0
3	MT	MAGNETIC TAPE	6	.20	3	3.3	0.00	0	0.00	:00.0
4	CR	CARD READER	1	.16	1	15.6	.19	3	.66	1:30:55.6
5	CP	CARD PUNCH	1	.12	1	11.8	0.00	0	0.00	:00.0
6	PR	PRINTER	1	.24	1	23.7	.32	3	.67	2:31:48.2
7	DF	DISK FILE	2	.03	.45	3.0	.03	1	.16	12:24.3
8	DP	DISK PACK	2	.17	2	8.4	0.00	0	0.00	:00.0

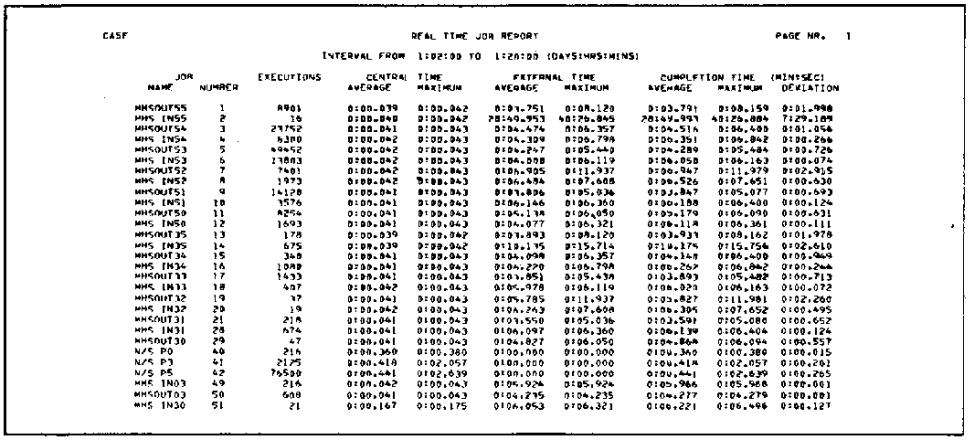
圖 3－53

CASE — COMMUNICATION NETWORK QUEUING REPORT — PAGE NR. 1

INTERVAL FROM 1:02:00 TO 1:20:00 (DAYS:HRS:MINS)

COMPONENT NUMBER	MODEL	MODULE CODE	MODULE NAME	UTILIZATION OF COMPONENT AVG	UTILIZATION OF COMPONENT MAX	QUEUE LENGTH DURING PERIOD OF AVG UTILIZATION AVG	AVG UTILIZATION MAX	AVG UTILIZATION DEV	MAX UTILIZATION AVG	MAX UTILIZATION MAX	MAX UTILIZATION DEV	TOTAL DELAY (HRS:MINS:SECS)
5	5953	T	TERMINAL	.01	.03	.00	0	.00	.00	0	.00	:00.1
7	5953	T	TERMINAL	.05	.08	.00	1	.00	.01	1	.01	1:04.0
10	5953	T	TERMINAL	.03	.05	.00	0	.00	.00	1	.00	:22.6
11	5953	T	TERMINAL	.03	.07	.00	0	.00	.00	1	.01	:42.9
12	5953	T	TERMINAL	.03	.06	.00	0	.00	.00	1	.00	:37.8
14	5953	T	TERMINAL	.04	.08	.00	1	.00	.01	1	.01	1:07.1
20	5953	T	TERMINAL	.03	.05	.00	0	.00	.00	1	.00	:22.6
21	5953	T	TERMINAL	.05	.09	.00	1	.00	.01	1	.01	1:13.6
22	5953	T	TERMINAL	.02	.04	.00	0	.00	.00	1	.00	:13.5
23	5953	T	TERMINAL	.20	.35	.05	1	.07	.19	3	.37	23:22.3
24	5953	T	TERMINAL	.02	.03	.00	0	.00	.00	0	.00	:09.9
25	5953	T	TERMINAL	.40	.70	.24	3	.54	1.65	36	6.71	21:41:13.3
32	5953	T	TERMINAL	.02	.03	.00	0	.00	.00	0	.00	:08.9
34	5953	T	TERMINAL	.02	.04	.00	0	.00	.00	1	.00	:14.0
35	5953	T	TERMINAL	.01	.03	.00	0	.00	.00	0	.00	:06.1
36	5953	T	TERMINAL	.03	.05	.00	0	.00	.00	1	.00	:25.2
37	5953	T	TERMINAL	.01	.03	.00	0	.00	.00	0	.00	:06.1
38	5953	T	TERMINAL	.01	.03	.00	0	.00	.00	0	.00	:06.1
40	5953	T	TERMINAL	.01	.03	.00	0	.00	.00	0	.00	:06.1
42	5953	T	TERMINAL	.12	.23	.02	1	.02	.07	1	.16	8:45.3
43	5953	T	TERMINAL	.12	.22	.01	1	.02	.06	1	.16	8:29.6
45	5953	T	TERMINAL	.03	.06	.00	0	.00	.00	1	.01	:33.6
46	5953	T	TERMINAL	.05	.07	.00	0	.00	.00	1	.01	:36.5
49	5953	T	TERMINAL	.19	.41	.06	1	.06	.29	4	.62	32:03.6
51	5953	T	TERMINAL	.04	.10	.00	1	.00	.01	1	.01	1:15.5
55	5953	T	TERMINAL	.04	.08	.00	0	.00	.01	1	.01	:57.4
56	5953	T	TERMINAL	.16	.35	.03	1	.04	.19	2	.35	24:09.4
57	5953	T	TERMINAL	.04	.08	.00	0	.00	.01	1	.01	1:00.4
58	5953	T	TERMINAL	.04	.08	.00	0	.00	.01	1	.01	:57.4
65	5953	T	TERMINAL	.01	.03	.00	0	.00	.00	0	.00	:06.1
68	5953	T	TERMINAL	.01	.03	.00	0	.00	.00	0	.00	:06.1
71	5953	T	TERMINAL	.08	.08	.01	1	.01	.01	1	.01	3:38.9
72	5953	T	TERMINAL	.26	.56	.01	1	.02	.27	4	.74	21:08.7
73	5953	T	TERMINAL	.26	.50	.02	1	.03	.31	5	.85	25:21.4
1	5953MX	T	TERMINAL	.08	.15	.01	1	.01	.03	1	.03	3:26.2
2	5953MX	T	TERMINAL	.39	.69	.24	3	.49	1.49	31	5.76	2:28:16.3
3	5953MX	T	TERMINAL	.04	.07	.00	0	.00	.00	1	.01	:48.3
4	5953MX	T	TERMINAL	.19	.33	.04	1	.06	.16	2	.29	20:31.8
6	5953MX	T	TERMINAL	.17	.36	.03	1	.05	.17	2	.21	16:02.6
8	5953MX	T	TERMINAL	.04	.06	.00	0	.00	.00	1	.00	:48.9
9	5953MX	T	TERMINAL	.12	.21	.02	1	.02	.05	1	.08	7:10.1
13	5953MX	T	TERMINAL	.38	.83	.29	3	.46	1.98	137	25.66	5:42:36.7
15	5953MX	T	TERMINAL	.10	.23	.01	1	.01	.07	1	.18	9:04.8
16	5953MX	T	TERMINAL	.10	.21	.01	1	.01	.05	1	.08	7:31.7
18	5953MX	T	TERMINAL	.19	.37	.06	1	.07	.27	3	.43	27:41.3
19	5953MX	T	TERMINAL	.08	.15	.01	1	.01	.07	1	.03	3:23.9
24	5953MX	T	TERMINAL	.30	.53	.19	2	.22	.60	9	1.58	1:07:57.6

圖 3－54

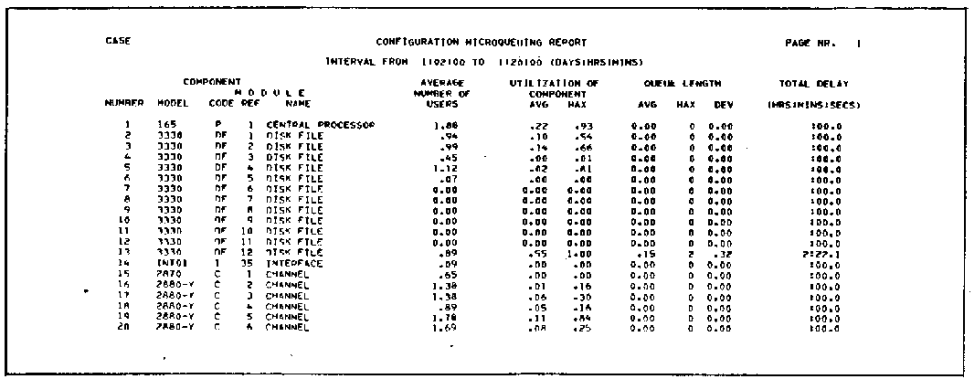


CASE　　REAL TIME JOB REPORT　　PAGE NR. 1

INTERVAL FROM 1:02:00 TO 1:20:00 (DAYS:HRS:MINS)

JOB NAME	NUMBER	EXECUTIONS	CENTRAL TIME AVERAGE	CENTRAL TIME MAXIMUM	EXTERNAL TIME AVERAGE	EXTERNAL TIME MAXIMUM	COMPLETION TIME AVERAGE	COMPLETION TIME MAXIMUM	(MINS:SEC) DEVIATION
MHSOUT55	1	A901	0:00.039	0:00.042	0:01.751	0:08.120	0:03.791	0:08.159	0:01.998
MHS IN55	2	16	0:00.040	0:00.042	28:40.953	48:26.845	28:49.993	48:26.884	7:29.189
MHSOUT54	3	23752	0:00.041	0:00.043	0:04.474	0:06.357	0:04.514	0:06.400	0:01.056
MHS IN54	4	6380	0:00.042	0:00.043	0:04.309	0:06.798	0:04.351	0:06.842	0:00.266
MHSOUT53	5	49452	0:00.042	0:00.043	0:04.247	0:05.449	0:04.289	0:05.484	0:00.726
MHS IN53	6	13803	0:00.042	0:00.043	0:04.068	0:06.119	0:04.058	0:06.163	0:00.074
MHSOUT52	7	7401	0:00.042	0:00.043	0:06.905	0:11.937	0:06.947	0:11.979	0:02.915
MHS IN52	8	1973	0:00.042	0:00.043	0:06.494	0:07.608	0:06.526	0:07.651	0:00.630
MHSOUT51	9	14128	0:00.041	0:00.043	0:05.896	0:05.936	0:05.937	0:05.977	0:00.693
MHS IN51	10	3576	0:00.041	0:00.043	0:06.146	0:06.360	0:06.188	0:06.400	0:00.124
MHSOUT50	11	8254	0:00.041	0:00.043	0:06.138	0:06.650	0:06.179	0:06.693	0:00.631
MHS IN50	12	1693	0:00.041	0:00.043	0:04.077	0:06.321	0:04.118	0:06.361	0:00.111
MHSOUT35	13	178	0:00.039	0:00.042	0:03.893	0:05.129	0:03.933	0:06.162	0:01.978
MHS IN35	14	675	0:00.039	0:00.042	0:13.115	0:15.714	0:13.154	0:15.756	0:02.610
MHSOUT34	15	348	0:00.041	0:00.043	0:04.098	0:06.357	0:04.140	0:06.400	0:00.949
MHS IN34	16	1048	0:00.041	0:00.043	0:04.220	0:06.798	0:04.262	0:06.842	0:00.244
MHSOUT33	17	1433	0:00.041	0:00.043	0:04.861	0:05.434	0:04.903	0:05.482	0:00.713
MHS IN33	18	497	0:00.042	0:00.043	0:06.078	0:06.119	0:06.120	0:06.163	0:00.072
MHSOUT32	19	37	0:00.041	0:00.043	0:06.785	0:11.937	0:06.827	0:11.981	0:02.260
MHS IN32	20	19	0:00.042	0:00.043	0:06.243	0:07.608	0:06.285	0:07.652	0:00.495
MHSOUT31	21	214	0:00.041	0:00.043	0:05.550	0:05.936	0:05.591	0:05.980	0:00.662
MHS IN31	22	674	0:00.041	0:00.043	0:06.097	0:06.360	0:06.139	0:06.404	0:00.124
MHSOUT30	23	47	0:00.041	0:00.043	0:04.827	0:06.050	0:04.868	0:06.094	0:00.557
N/S P0	40	214	0:00.360	0:00.380	0:00.000	0:00.000	0:00.360	0:00.380	0:00.015
N/S P9	41	2125	0:00.418	0:02.057	0:00.000	0:00.000	0:00.418	0:02.057	0:00.261
N/S P8	42	76599	0:00.441	0:02.639	0:00.000	0:00.000	0:00.441	0:02.639	0:00.265
MHS IN03	49	214	0:04.042	0:05.043	0:05.924	0:05.924	0:05.966	0:05.968	0:00.001
MHSOUT03	50	608	0:00.041	0:00.043	0:04.235	0:04.235	0:04.277	0:04.279	0:00.001
MHS IN30	51	21	0:00.147	0:00.175	0:06.063	0:06.321	0:06.221	0:06.496	0:00.127

図3－55



CASE　　CONFIGURATION MICROQUEUING REPORT　　PAGE NR. 1

INTERVAL FROM 1:02:00 TO 1:20:00 (DAYS:HRS:MINS)

COMPONENT NUMBER	MODEL	MODULE CODE	MODULE REF	MODULE NAME	AVERAGE NUMBER OF USERS	UTILIZATION OF COMPONENT AVG	UTILIZATION OF COMPONENT MAX	QUEUE LENGTH AVG	QUEUE LENGTH MAX	QUEUE LENGTH DEV	TOTAL DELAY (HRS:MINS:SECS)
1	165	P	1	CENTRAL PROCESSOR	1.06	.22	.93	0.00	0	0.00	100.0
2	3330	DF	1	DISK FILE	.94	.18	.56	0.00	0	0.00	100.0
3	3330	DF	2	DISK FILE	.99	.14	.66	0.00	0	0.00	100.0
4	3330	DF	3	DISK FILE	.45	.00	.01	0.00	0	0.00	100.0
5	3330	DF	4	DISK FILE	1.12	.02	.41	0.00	0	0.00	100.0
6	3330	DF	5	DISK FILE	.07	.00	.00	0.00	0	0.00	100.0
7	3330	DF	6	DISK FILE	0.00	0.00	0.00	0.00	0	0.00	100.0
8	3330	DF	7	DISK FILE	0.00	0.00	0.00	0.00	0	0.00	100.0
9	3330	DF	8	DISK FILE	0.00	0.00	0.00	0.00	0	0.00	100.0
10	3330	DF	9	DISK FILE	0.00	0.00	0.00	0.00	0	0.00	100.0
11	3330	DF	10	DISK FILE	0.00	0.00	0.00	0.00	0	0.00	100.0
12	3330	DF	11	DISK FILE	0.00	0.00	0.00	0.00	0	0.00	100.0
13	3330	DF	12	DISK FILE	.80	.55	1.00	.15	2	.37	2122.1
14	INT01	I	35	INTERFACE	.09	.00	.00	0.00	0	0.00	100.0
15	2870	C	1	CHANNEL	.65	.00	.00	0.00	0	0.00	100.0
16	2880-Y	C	2	CHANNEL	1.38	.01	.16	0.00	0	0.00	100.0
17	2880-Y	C	3	CHANNEL	1.38	.06	.30	0.00	0	0.00	100.0
18	2880-Y	C	4	CHANNEL	.89	.05	.14	0.00	0	0.00	100.0
19	2880-Y	C	5	CHANNEL	1.78	.11	.44	0.00	0	0.00	100.0
20	2880-Y	C	6	CHANNEL	1.69	.08	.25	0.00	0	0.00	100.0

図3－56

B. *CSS

(1) 概　要

CSS(Computer System Simulator)は，IBM社が提供しているコンピュータ・システム専用のシミュレーション・システムである。

CSSは，GPSSとよく似た概念のシミュレータではあるが，汎用ではなく，コンピュータ・システムにのみ適用可能であるという点が異なっている。

＊CSSについてはIBM社の"CSS General informatin manual"を参照させて頂いた。

したがって，コンピュータ・システム特有のtape units，disk files，communication lines，terminalsとか，プログラミング用の命令語などのタームがCSS言語中に備えてある。

CSSによれば，360/370の全シリーズのモデル化が可能で，新しい機器や特殊システムなどのモデル化にも比較的容易に適合できると言うことである。

CSSを利用するにあたって，利用者はまず，機器構成，システムへの入力，プログラム・ロジック，タイミング情報をCSS言語により記述する。次に，システム・オペレーションの状態がシミュレートされ，システム・パフォーマンス向上に役立つようなレポートが出力される。その結果を検討し，必要があればモデルを変更して再びシミュレートすることになるわけである。

図3-57にその概要を示す。

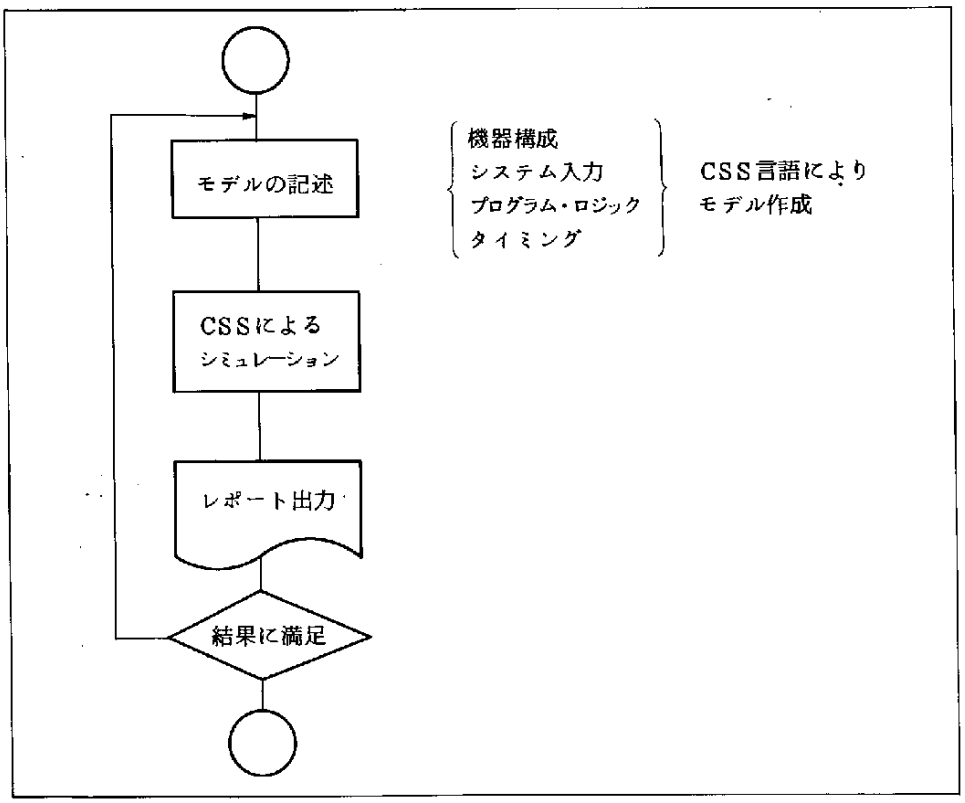


図3－57　CSSによるシミュレーション

CSS にはオートマティック・デザインの機能がないので，利用者は自分のアプリケーション・プログラムだけでなく，オペレーティング・システムの動作についても，かなり習熟する必要がある。したがってCSS によりモデルを記述するには，かなりの調査が必要となり，システム・オペレーションに関する疑問の多くは，モデルができあがるまでに解決してしまう事もあり得る。

このように，シミュレーションのためにシステム・ロジックを厳密に調査することは，システム・デサイナにとっても非常に有用であるし，システム・オペレーションの理解にも役立つと言っている。

(2) システム・モデルの記述

利用者が，システム・モデルを記述する内容は，次の3つからなっている。

①システム構成の記述（I／O スピードも含む）

各デバイス毎に I／O スピード，結合状態などの情報と共に，システム構成を記述。

②プログラムに関する記述

プログラム・ロジック，処理時間，統計的な情報などを記述する。

③システムの運用されるエンヴィロンメントの記述

これは①，②の記述に暗に含まれてはいるが，各ターミナル毎のメッセージ率とか，一つの入力装置から起動されるジョブ・ストリームの数などに関する情報である。

① システム構成の記述

CSS でシミュレートできる装置は以下のようなものである。

- CPU（マルチ・プロセッサも可）
- I／O 装置と結合用のチャネル各種
- 通信回線（contention，polling）
- 端末装置各種

したがって下記のような構成のモデルが記述できる。

・シングル・プロセッサのカード or テープ・システム

・ディスク・ベース・システム

・マルチ・プロセッサ・システム

・テレ・プロセシング・システム

・上記の組合せ

また，オペレーション形態としては次のような形態があつかえる。

・シークエンシャル処理

・マルチ・プログラミング

・リアル・タイム処理

・タイムシェアリング処理

以下にシステム構成記述の例を示す。

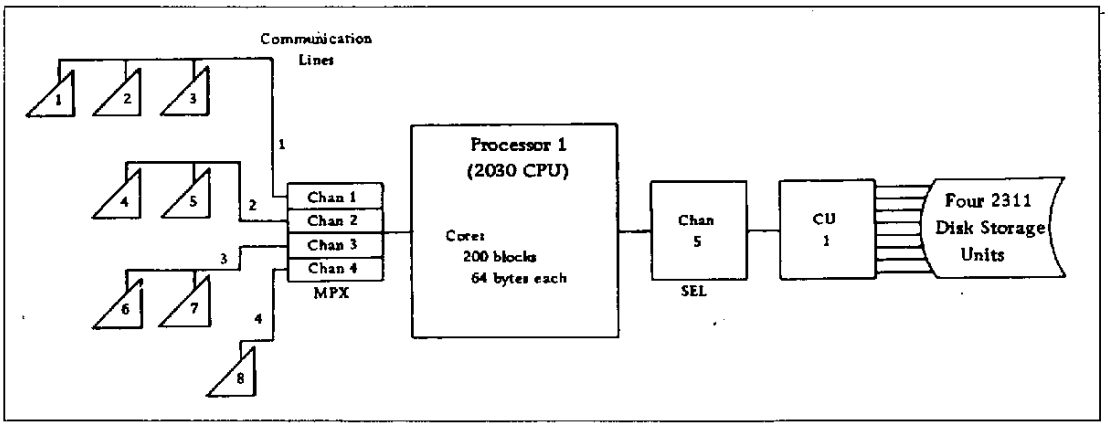


図３－５８　Data collection systemの例

Unit Number	Unit Type	Unit Characteristics
	SYSTEM	PR1,MPX1,CU1
1−4	LINE	15
1−4	CHANNEL	1,, 16, 1
5	CHANNEL	1,, 667
1	RESOURCE	200,,, 64
1−4	2311	, 1, 156000, 25000
1−8	TERMINAL	, 1, 2
	RATE	1/70, 2−4/50, 5−6/85, 7/94, 8/38
	RATH	1/1−1, 2/2−2, 3/3−3, 4/4−4, 5/5−1, 6/5−2
	INPATH	1−3/1, 4−5/2, 6−7/3, 8/4

図３－５９　図３－５８のCSS言語による記述

② プログラムに関する記述

CSSではプログラム記述のために41種の命令を用意している。この命令を用いて，利用者は，自分のアプリケーション・プログラムとコントロール・プログラムの２クラスに分けてプログラム構造を記述する図3−58に示したようなオンラインのデータ収集システムのプログラムは単純化すると図3−60のようになるが，これをCSS言語で記述した例が図3−61である。

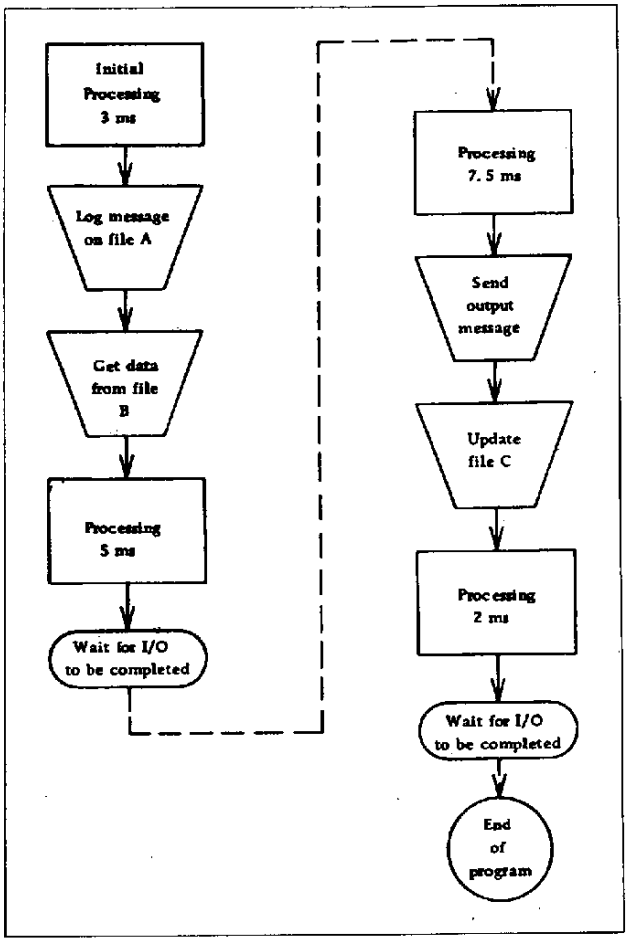


図３－６０　アプリケーション・プログラムの例

```
AP1     PROCESS      30     Initial processing
        WRITE      CPROT1   Log message to file A
        READ       CPROT3   Access file B
        PROCESS      50     Overlap I/O processing
        WAIT                Suspend processing until I/O complete
        PROCESS      75     Unoverlapped processing
        SEND       MPROT1   Send output
        WRITE      CPROT3   Update file C
        PROCESS      20     Overlap I/O and processing
        WAIT                Wait for I/O
        RETURN              End of program
```

図３－６１　CSS 言語によるコーディング

CSSでは，コントロール・プログラムもCSS言語を用いて利用者が記述するが，図3−62はスケジューラの部分を表したフローで，図3−63は，CSS言語でコーディングした例である。

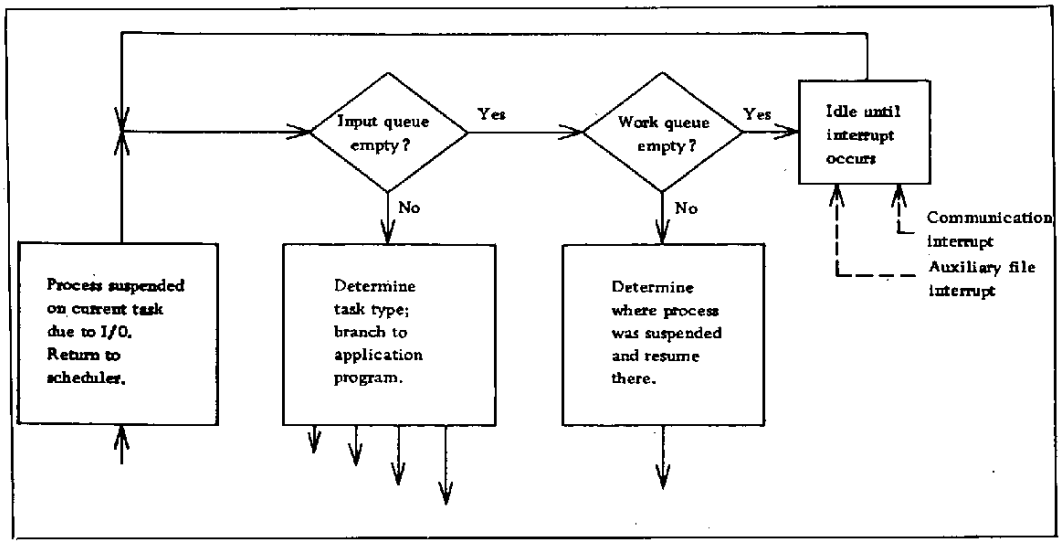


図3−62　ジョブ・スケジューラ

```
SCHEDL   TEST     NTQ1,0,*+3      Is input queue(TQ1)empty?
         QMOVE    TQ1,,CTASK      No-get first queued task.
         ENTER    REFTL(1,TTY)    Branch to program based on type.
         TEST     NTQ2,0,*+3      Is work qucue(TQ2)empty?
         QMOVE    TQ2,,CTASK      No-get first queued task.
         MOVDUB   DDUB,CDUB       Branch to resume process.
         IDLE                     All queues empty.
```

図3−63　CSS言語によるコーディング

(3)　出力レポートの指定

CSSによる統計情報の出力には，システムが自動的に出力してくれるものと，利用者が実行中に収集しておいて出力するものの2種類がある。

システムが自動的に収集し出力してくれる情報には，以下のようなものがある。

- 入力情報に関する統計表
- 各装置の利用率（図3−65）
- システム内部のQueueに関する統計（図3−66）
- コアのブロックなどのシステム・リソースの統計（図3−67）
- アクティビティの統計（図3−68）

利用者が定義する出力としては，例えば，各プロセスの平均処理時間，特定のイベントの生起回数，メッセージのレスポンス・タイムの分布など多種の指定が可能である。

MESSAGE RESPONSE TIME. TIME IN MILLISECONDS.

ENTRIES IN TABLE	MEAN ARGUMENT	STANDARD DEVIATION
200	1625.049	539.000

UPPER LIMIT	OBSERVED FREQUENCY	PERCENT OF TOTAL	CUMULATIVE PERCENTAGE	CUMULATIVE REMAINDER
0	0	0.000	0.0	100.0
500	0	0.000	0.0	100.0
1000	51	25.500	25.5	74.5
1500	74	37.000	62.5	37.5
2000	38	19.000	81.5	18.5
OVERFLOW	37	18.500	100.0	0.0

AVERAGE VALUE OF OVERFLOW 2297.729

図3−64　Distribution table

PROCESSOR NUMBER 1 CURRENTLY BUSY

TOTAL PROCESSOR UTILIZATION .8942

UTILIZATION DUE TO CHANNEL INTERFERENCE .0559

CHANNEL	TYPE	UTILIZATION	TOTAL READS	TOTAL WRITES
1	M 1	.2224	48	0
2	M 1	.2529	54	0
3	M 1	.2410	52	0
4	M 1	.2249	48	0
5	S	.5615	1003	200

DEVICE TYPE	NUMBER	UTILIZATION	TOTAL READS	TOTAL WRITES
2311	1	.6383	257	50
	2	.6571	271	52
	3	.5652	221	54
	4	.5946	254	44

図3−65　Equipment utilization

QUEUE NUMBER	CURRENT CONTENTS	MAXIMUM CONTENTS	AVERAGE CONTENTS	TOTAL ENTRIES	AVG. TIME IN Q %SEC□	DEVIATION FROM MEAN CONTENTS
1	1	10	1.736	202	0.370	2.160
2	0	6	1.239	591	0.090	1.324
3	0	0	0.000	0	0.000	0.000

図3－66　Queue statistics

NUMBER	CAPACITY	CURRENTLY AVAILABLE	MINIMUM AVAILABLE	AVG. IN USE	UTILIZATION	AVG. TIME IN USE %SEC□
1	50	47	17	10.0626	.2012	0.4329

図3－67　Resource statistics

TERMINAL NUMBER	MESSAGES GENERATED	MESSAGES TRANSMITTED	MESSAGES RECEIVED
1	48	0	0
2	55	0	0
3	52	0	0
4	49	0	0

図3－68　Terminal activity

C. *AMAP

AMAP（Advanced Multiprogramming Analysis Procedure）は，IBM 360/370 OSのもとで実行されるユーザ・ジョブのシステム内部における動作履歴をイベント方式で収集し，そのワーク・ロードを解析すると共に，そのワーク・ロードの他のIBM機種におけるパフォーマンスをシミュレートし予測するシステムである。

＊AMAPはIBMの社内利用に限られ，紹介されている文献もごく少ない。
本節では，「情報処理 VOL13, No.11」橋本氏（東レ）の論文，他を参考にさせて頂いた。

AMAP による測定・分析結果は次のような目的の為に利用できる。

① ユーザ・ジョブによるワーク・ロードとシステム・パフォーマンスの関係を分析し，ユーザ・プログラムの効率向上のための資料を得る。

② OS 資源の稼動状況を分析して OSの利用法の改善を図る。

③ 得られた基礎データをもとにモデリングにより，新しい構成のシステムの効率を予測する。

AMAPは，これらのことを行うために選定されたユーザ・ジョブの処理中のトレース・データを収集し以下のような分析を行うために記録しておく。

① ラン・アナリシス（Run Analysis）

システムの処理状況を明らかにし，性能改善のポイントをシステム全体の観点より分析する。

② ジョブ・アナリシス（Job Analysis）

対象ジョブごとの分析で，それぞれのジョブ処理におけるボトルネックを抽出し性能改善の資料を得る。

③ モデリング（Modeling）

実際に収集したジョブの処理データにもとづいて，システム構成の一部（または全部）を変更した場合の同一ジョブの動作をモデルによってシミュレートし新システムの処理性能を予測する。

(2) AMAP の機能（図 3-69）

AMAP は，実際にジョブが処理されている状態を追跡し，その動きの時系列データを磁気テープ（トレース・テープ）上に記録する。このようにして収集されたデータは各プログラム固有の動作履歴とシステムの各機能の使用状況を特徴づけるものである。

データ収集ランのユーザ・ジョブの処理時間に与える影響は最大限５％といわれている。

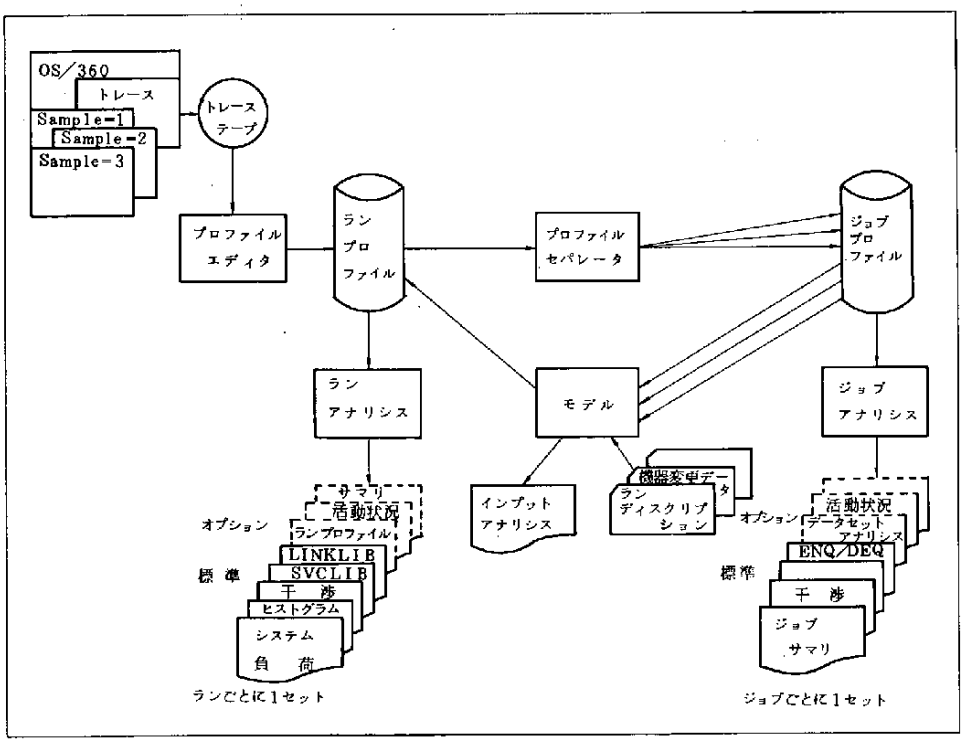


図3－69　AMAPの機能図

作成されたトレース・テープからプロファイル・エディタによりラン・プロファイルを作成する。このラン・プロファイルはラン・アナリシスのための入力となる。

ジョブ単位の分析を行なうためには，このラン・プロファイルから，プロファイル・セパレータにより各ジョブごとのジョブ・プロファイルを作成する。ジョブ・プロファイルは，ジョブ・アナリシスへの入力になると共に，モデルへの入力ともなっている。

ラン・プロファイル，ジョブ・プロファイルを分析するために次の3つの手続きが用意されている。

① ラン・アナリシス

ラン・プロファイルを入力として次のような項目の分析結果を出力する。

- システム負荷（System Loading）
- 各ジョブステップの開始・終了時間
- 各ジョブ，ジョブステップごとに，経過時間，CPU稼動時間，CPU待ち時間,すべてのデータセットの物理的位置，入力・出力の回数および時間
- 入出力装置ごとにデータセット相互間の干渉度
- 主記憶装置上にないスーパバイザ・モジョールの使用頻度

② ジョブ・アナリシス

ジョブ・プロファイルを入力として次のような項目の分析結果を出力する。

- 処理時間
- 各データセットの参照頻度
- データセット間の干渉度
- データセット間の同時アクセス頻度
- 主記憶装置上にないシステム・ライブラリーの使用頻度
- その他処理状況の詳細な分析

ジョブ・アナリシスの結果はたとえば次のように利用される。

データセット間の干渉度の高いものは異なる入出力装置上に分離する。データセット間の同時アクセス頻度が高いものはチャネルを分離する等。

③ モデリング

システム構成やジョブ構成を変えた模擬システムのモデルを与え，ジョブ・プロファイルを入力すると模擬システムの動作を予見して，そのラン・プロファイルを作る。このラン・プロファイルをラン・アナリシ

スで分析することによりモデル化されたシステムの特性が把握でき，業務に適合したシステム構成を選択することができる。

モデリングの際，変更できるのは以下のようなものである。

- リーダ，イニシェータ，ライタの数
- リーダ，イニシェータへのジョブの順序
- ジョブおよびジョブ・ステップの優先度
- ジョブ・イニシェータ，ジョブ・ライタのクラス等
- 主記憶容量
- 処理装置
- チャネル，コントローラ，I／O 装置の型，数等
- IOS のキューイング・テクニック
- データ・セットの収容装置
- データ・セットのブロッキング・ファクタ
- スーパバイザ・モジュールの常駐，非常駐

この他，テレ・プロセシングのモードで稼動するシステムに対しても各種の分析が可能になっている。

(3) AMAP による測定分析手順（図3－70）

AMAPはIBM社のシステム・エンジニアの管理のもとに使用されるため，使用法の細部については明らかでないが大体次のような手順を踏む。

① AMAP ランの準備―対象ジョブの選定

対象ジョブの選定は次のような規準で行なう。

- 主要なアプリケーションを代表するジョブを選ぶ
- プログラム用言語，プログラムに変化をもたせ，数分のジョブを数多く用意する

一般に数分のジョブから得られるデータも数十分のジョブから得られるデータも分析のための情報量としてはさほど変わらないと言われているので処理時間が短く，しかも業務の特性を代表するようなジョブを適切に選ぶことが経済性の面から要求される。

② トレース・ラン ── データの収集

トレース・プログラムを通常のOSジョブとして実行させ，同時に選定されたジョブを流し，システムの動作を磁気テープに記録する。

③ 編集ラン ── ラン・プロファイルの作成

④ ジョブ・プロファイルの作成

⑤ ラン・アナリシスの実施

⑥ ジョブ・アナリシスの実施

⑦ モデル化 ── システムの再設計

⑥までの手順で得られた分析結果にもとづいてシステムのボトルネックを改善するようなシステム構成の変更を行なう。あるいはあらかじめシミュレートしたいシステム構成を作成する。作成された新しい模擬システムについてAMAPによるモデル化を行ない，すでに作成されているジョブ・プロファイルを入力として新しいラン・プロファイルを作成する。

⑧ ラン・アナリシスによる再分析

模擬システムの分析を行なう。さらに種々のモデル化を行ないたいときは(7)，(8)の手順を繰り返す。

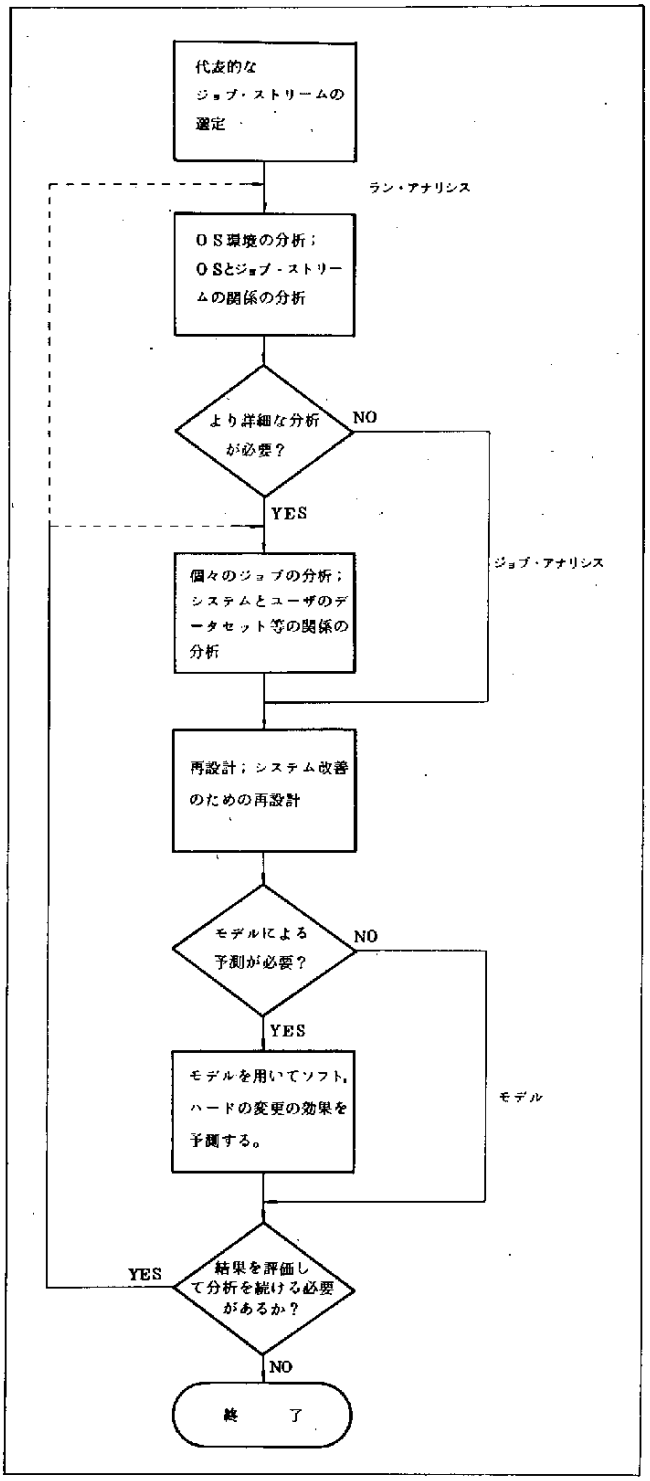


図３－７０　AMAPの利用手順

参 照 文 献

1) " Computer peformance analysis : Applications of accounting data "

R. Watson

NASA report No. R-573-NASA/PR May '71

2) Systems performance and evaluation — Past, present, and future

C. Dudley Warner '72 SJCC P959-P964

3) Choosing a system stethoscope : Larry E. Hart 他

Computer Decisions '71 NOV P20 - P23

4) Interpreting the results of hardware system monitor

J. S. Cockrum 他 '71 SJCC

4．タイムシェアリング・システムの動作解析と評価

4. タイム・シェアリング・システムの評価

タイム・シェアリング・システム評価のむずかしさは2つあると言われている。1つは，負荷が不安定で再現性の無い事。他の1つは，評価量としての決め手がつかめない点だといわれている。一般のコンピュータ・システムに於ても定量的な評価は難かしいが，タイム・シェアリング・システムに於てはより多くの困難な問題をかかえ，多方面から手さぐりで進めているのがむしろ現状の様に見受けられる。こうした現状をふまえて，この章ではタイム・シェアリング・システムを中心とする大型コンピュータ・システムの評価の問題点，特にワークロードの問題を軸に研究の方向を概観した後，現実にどの様なアプローチがなされているのかをケーススタディーを含めて解説し，最後にコンピュータ・システムの解析的研究にかかわる仕事の流れについて触れている。

4.1 タイム・シェアリング・システム評価の問題点（ワークロードの問題を中心に）

タイム・シェアリング・システム（ＴＳＳ）は，1950年代の後期に，ＭＩＴに於て研究が開始されて以来多方面に渡る研究開発がおこなわれ，数多くの評価に関連する報告も発表されてきたが，まだ多くの問題が残されたままの状態にあると言える。そのうちの１つがワークロードである。この問題は，長い間，むしろ，なおざりにされて来た感がある。それは決して，この問題が重要でないと思われていた為ではなく，むしろ，その困難さの故と言うことも出来る。

4.1.1 ワークロードの問題

現実のタイム・シェアリング・システムを評価する際に，どの様なワークロードのもとで行うかは重要な問題である。特にＴＳＳの場合には，バッチ処理システムの場合ほど手軽にワークロードを得る事は出来ない。実際，何十台もの端末を使用して必要なワークロードを得るのは人手もいるであろうし，またそれを随意にコントロールする事も大変な事である。ワークロードの問題は，言わばデータ測定状況の設定と言い換えても良いかもしれない。これに対するごく自然な方法としては，実際にシステムが稼動している状態を測定する事が一般的におこなわれている。しかし，この方法では限られた状態の把握のみになり易く，またその状態を自由に変更する事は出来ない。そこで，巾広い状態に対するワークロードを人工的に作り出す２・３の試みがなされている。その１つを4.2.2で紹介するが，ＲＣＡのＴＳＯＳの評価に用いられたものである。これは，端末ユーザが実行するプログラムに似た振舞いをする擬似的なプログラム（Pseudo Program と呼んでいる）を使用する方法である。この場合，Pseudo Program 自身が非常にシンプルなものである為に，十分端末ユーザに近い動作をするかどうかで難点が無いとも言えないが，プログラム自身の動作をコントロールするパラメータが自由に変えられ，色々なワークロードを作り出せる点で特徴がある。

もう1つの例はＭＩＴでＭＵＬＴＩＣＳの評価を行なう為にGreenbaumらが開発したロードシミュレータである。

これはＰＤＰ－８を電話回線を通してＧＥ－６４５に接続し，同時に12人の端末Userを Simulate する事が出来る様にしたものである。

このシステムでは端末ユーザの動作を記述する言語を持っており，あらかじめ，ＳＣＲＩＰＴを与えておけば，多様なユーザ・ビヘイビアーをシミュレート出来る。いくつかのＳＣＲＩＰＴＳが開発されているが，そのうち，端末からＦＯＲＴＲＡＮプログラムをタイプインしているユーザのSimulateしたものがよく使われているそうである。

ＭＩＴのSaltzer は，この様なロード問題に対して「オペレーティング・システムの微妙な変化が，パホーマンスに及ぼす影響を調べる問題については，与えるロードが不均質であると，その効果が十分に評価出来ない」と述べている。

シミュレーションに於けるワークロードの問題も，現実システムを対象とする評価とまったく同じ意味で重要な問題である。広い意味で，解析モデルも一種のシミュレーションと考えるなら，この場合のワークロードはシステムに対するジョブの到着分布と各ジョブの実行時間，等によって与えられる。
しかしながら，これらのワークロードは取扱いの簡単な，確率分布で与えられるのが常である。Denning が，「実験的なデータがないという事が，計算機システム内でのインタラクションの性質を説明をする解析的なモデルの開発をさまたげてきた」と述べている様に，この様なワークロード設定には現実的な裏付けが欠けていたと言える。イベントタイプのシミュレーションに於ても同じ様な傾向がみられていた。しかし，最近になってから，計算機システムの内部データの測定技術の進歩に伴ってワークロードの性格の分析評価の質も向上し，分析のアプローチも多様になって来た。しかし，ワークロードの特性をどんな量で表現したら良いかと言う問題が完全に解決されたわけでない。

ケーススタディーではタスク動作の分析を行った例も紹介してあるが，これもこの問題に対する一つのアプローチである。

ページング・システムにおけるプログラム動作をアドレス・トレースを行って分析する研究もさかんに行われ始めている。また，プログラム動作をＳＶＣ（Super Visor Call）とコンピュート・タイムの分布で捕えた例も報告されている。

ワークロードの問題に対するこれらの研究はやっと本格的に始まったばかりである。むしろ，これからの研究分野であると言えよう。この様な研究とともに，一方では現実的なデータをもとにしたシミュレーションの例が数多く報告されて来た。4.2.3にはその１つの例としてＲＣＡのWinogradらに依って行われたシミュレーションをケーススタディーとして紹介している。

シミュレーションに於けるワークロードの問題は，シミュレーション・モデルそのものと密接なかかわり合いを持っている。現実に，どの場合にもモデル自身の中に組込まれてしまっているのが普通である。この事は，ジョブロード自身のモデル化が必要な事を意味している。しかも，この問題は前に述べたワークロードの特性表示そのものであり，シミュレーションの中にもワークロードの問題は依然同じ形で残されていると言える。

具体的にシミュレーション・モデルにワークロードを組込むには２つの代表的な方法が行われている。その１つはワークロードの特性を統計的な分布で与える方法で，一般には，平均値や分散等の統計量で決められる疑似乱数として与えられる。これはいわゆるモンテカルロ・シミュレーションと呼ばれる方法で，一般に良く用いられる方法である。もう１つの方法は，トレース・ドリブン（Trace-driven）と呼ばれる方法である。これについては，テキサス大学のSherman らによって行われた例がある。この評価の目的はスケジューリング・アルゴリズムの比較検討であったが，トレース・ドリブン型のシミュレーション・モデルを使っている。シミュレーション・モデルのワークロードとして，実際にシステムが稼動している際に，ソフトウエア・モニターでディティルなイベント系列を磁気テープに取り出し，直接シミュレーション・モデルをドライブするワークロード・データとして使っている。これは，汎用シミュレーション言語であるＳＩＭＳＣＲＩＰＴのExsogenous Event の取扱いと全

く同じ考え方に基づくものである。

4.2 評価の実例

この節では，タイム・シェアリング・システム評価に関する多様の研究成果の中で，4.1でのべたワークロードの問題を中心的にとらえたと思われる代表的なものを4件紹介する。

(1) Murty ParuPvdi, Joseph Winograd に依る，
「タイム・シェアリング・システムに於けるインタラクティブ・タスクの動作」

(2) W·M·DE Meis と N·Weizer による，
「デマンドページ・タイム・シェアリング・システムの評価と解析」

(3) J·Winograd, S·J·Morgantein, R·Herman による，
「バーチャルメモリー，タイムシェアード，デマンドページ・オペレーティング・システムのシミュレーション」

(4) H·D·Schwetman, J·C·Brown による，
「コンピュータ・システムの実験的研究」

4.2.1 タスク動作の解析

大型コンピュータ・システムのもとでのプログラム動作を解析しその結果をまとめている。

一口にプログラム動作と言っても，その特性をとらえる視点はいろいろと設定出来る。しかも，どれが一番適当であるかは一概に決められないむずかしい問題である。

この報告では，タイム・シェアリング・システムに於けるインタラクティブ・タスクの動作を，

① コンピュート・タイム

② スィンク・タイム（Think time）

③ ページフォールトの発生頻度

④ ページフォールト間のコンピュート・タイム

等の特性から明らかにしようと試みている。

UNIVAC Series 70のオペレーティング・システムVMOS（Virtual Memory Operating System）を対象としてインベント・タイプのソフトウエア・モニタCOSMOSを使いデータ測定を行っている。

A. ソフトウエア・モニタについて

ＩＢＭ360/67のタイム・シェアリング・システムに組込まれたＳＩＰＥと呼ばれるソフトウエア・モニタについては第2章でも述べてある。この実験に使われたCOSMOS（Continuous Software Monitoring System）もこれと似た考えで作られている。この様なイベントタイプのソフトウエア・モニタはサンプリング・タイプが一定時間間隔でデータを集めるのに対して，Ｉ／Ｏ割込み，タスク・アクティベイション，スーパバイザ・コール(ＳＶＣ）などのイベントが発生すると，その都度情報を取り出す方式を採っている。その為，オペレーティング・システムの各機能の中にイベント・データを取り出すプログラムが組込まれており，コントロールが渡ると自動的にイベント情報を作成し，バッファー（一般にメインメモリーに設けられている。）に貯め込んで行き，バッファーが満たされると外部記憶装置（磁気テープとかドラム）に書き出す仕組になっている。しかし，どのような場合にも全てのイベントを対象としていては，ソフトウエア・モニタのオーバ・ヘッドが大きくなってしまうし，後で行う解析の手間も大変なので，普通，イベントの種類を指定し，取り出すデータを制限する事が出来る様になっている。

次の図4.1はCOSMOSが各イベントごとに作り出す情報の内容を示したものである。

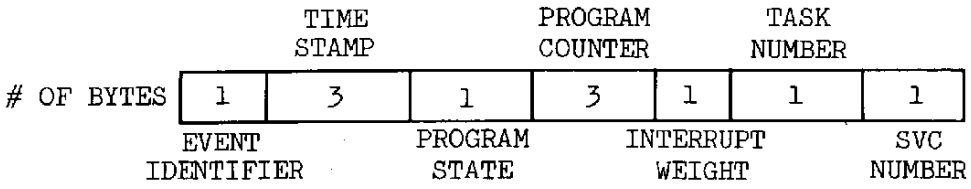


図4－1　COSMOSのイベント情報

B. データの測定

データの測定は通常のユーザがこのシステムを使っている状態で行われた。1回の測定は連続して2時間以上行い，なるべく頻繁にシステムが使われている時を選び，少なくとも10端末以上がアクティブになっている様にした。(端末は30台までである。)この中から適当なデータ・グループを選び解析を行っている。各データ・グループにA，B，C，Dの名前が付けられているが，これは後に示す，結果の表やグラフの記号と対応している。尚，各データ・グループに於ける端末ユーザの平均会話時間はA，B，C，Dの各々60分，86分，77分，66分である。

C. 結果の解析

一般にソフトウエア・モニタで得たオリジナルな情報は，前の図4－1にも示してある通り，非常にシンプルなものである。従って，これから必要な情報，例えば，Think time，Compute time などと言った量を求める為には，オリジナル・テープをリダクションするプログラムが必要である。ソフトウエア・モニタでオリジナル・データを得る事は手軽であるが，本当に必要な情報を得るにはかなりの手間を必要とする。(この報告では，特にリダクション・プログラムについては触れていない。)

① Think time

Think time の定義はあまり統一されておらず，いくつかの異った定義がされている。ここでは，オペレーティング・システムの側から測る意味もあって，端末に対してRead 要求を出してから，インプットが完全に終了するまでの時間をThink time と呼んでいる。各データ・グループについてThink time の分布が次の図4－2と表4－1に示されている。

表4－1　THINK TIME (SECONDS)

	OBSERVED DATA				NORMALIZED DATA				
	A	B	C	D	A	B	C	D	SUM
#INTER-ACTIONS	1075	1522	1543	1530	1370	1370	1370	1370	1370
MEAN	25.6	18.0	23.1	20.8	11.5	9.8	12.6	10.8	11.2
MEDIAN	10.3	8.6	10.6	9.8	9.5	8.2	10.0	9.4	9.3
MODE	3.5 (8.9%)	3.5 (8.3%)	6.5 (7.3%)	5.5 (8.0%)	3.5 (9.9%)	3.5 (9.3%)	6.5 (8.1%)	5.5 (9.0%)	3.5 (8.3%)
STANDARD DEVIATION	68.8	39.7	47.7	48.5	9.8	8.0	10.5	8.0	9.2

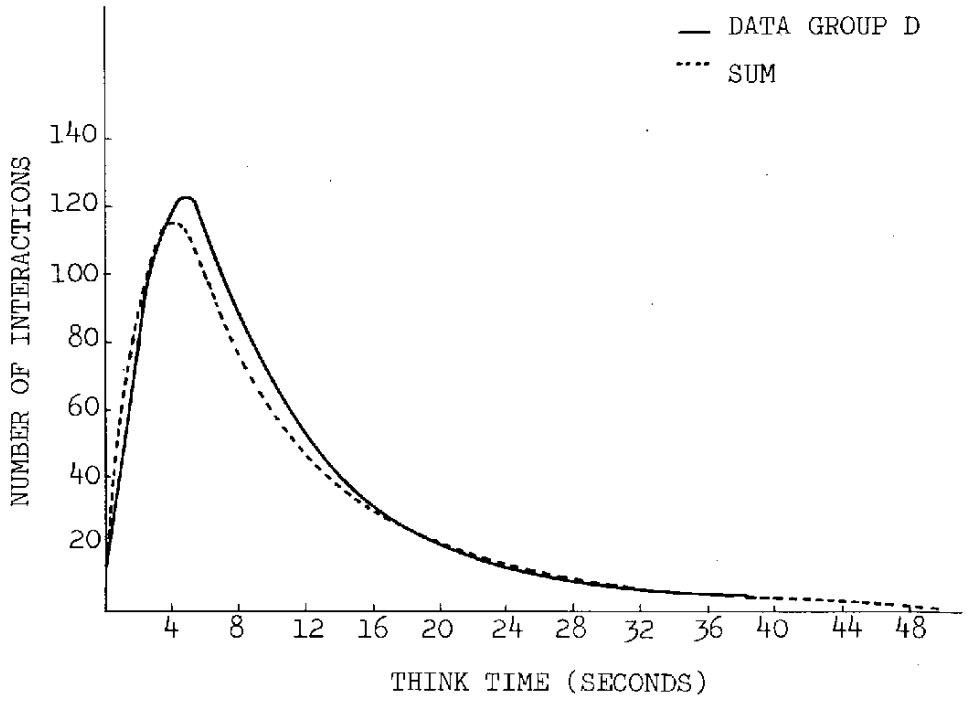


図4－2　Think time の分布

表４－１で左側のデータは観測データであるが右側のデータは左側の各データ・グループについて，サンプルを変量（この場合Think time）の小さい順に並べ，始めから90%のサンプルを選び，各統計量を計算したものである。比較をし安くする為にデータ・グループDのサンプル数を基準にして正規化（グラフ上の面積が等しくなる様に）してある。また，インタラクションと言っているのは，１つのタスクが同じ端末に対してREAD要求を続けて出す間を指してこう呼んでいる。（端末に対するWRITEは含んでいない。）

表４－１のデータからみると，データ・グループDの場合，全部の観測データの平均が20.8 秒であるのにその90%のデータの平均値が10.8 と半分近く小さくなっている。この事は，全データのわずか10%の中に極端に長いThink time を持つデータが含まれている事を示している。この為に観測データの標準偏差に比べ90%データの方の標準偏差は極端に小さくなっている。

② コンピュート・タイム（Compute time ）

これは各インタラクション当りのコンピュート・タイムの分布を調べたもので，その結果が表４－２及び図４－３に示されている。

表のデータに対しては Think time の時と同じ正規化を行っている。このデータについても，一部データに非常に大きな値を持つものがある事が認められる。

表4－2　COMPUTE TIME PER INTERACTION (MILLISECONDS)

	OBSERVED DATA				NORMALIZED DATA				
	A	B	C	D	A	B	C	D	SUM
#INTER-ACTIONS	1051	1498	1514	1500	1344	1344	1344	1344	1344
MEAN	179.2	287.4	346.2	340.2	45.6	45.1	56.3	38.2	46.3
MEDIAN	34.3	27.2	35.8	25.1	28.2	20.8	29.6	19.8	24.1
MODE	9.5 (6.0%)	9.5 (5.7%)	9.5 (5.4%)	9.5 (5.6%)	9.5 (6.7%)	9.5 (6.3%)	9.5 (6.0%)	9.5 (6.3%)	9.5 (6.3%)
STANDARD DEVIATION	924.1	1943.1	1827.5	3601.8	45.7	50.7	68.1	42.2	53.0

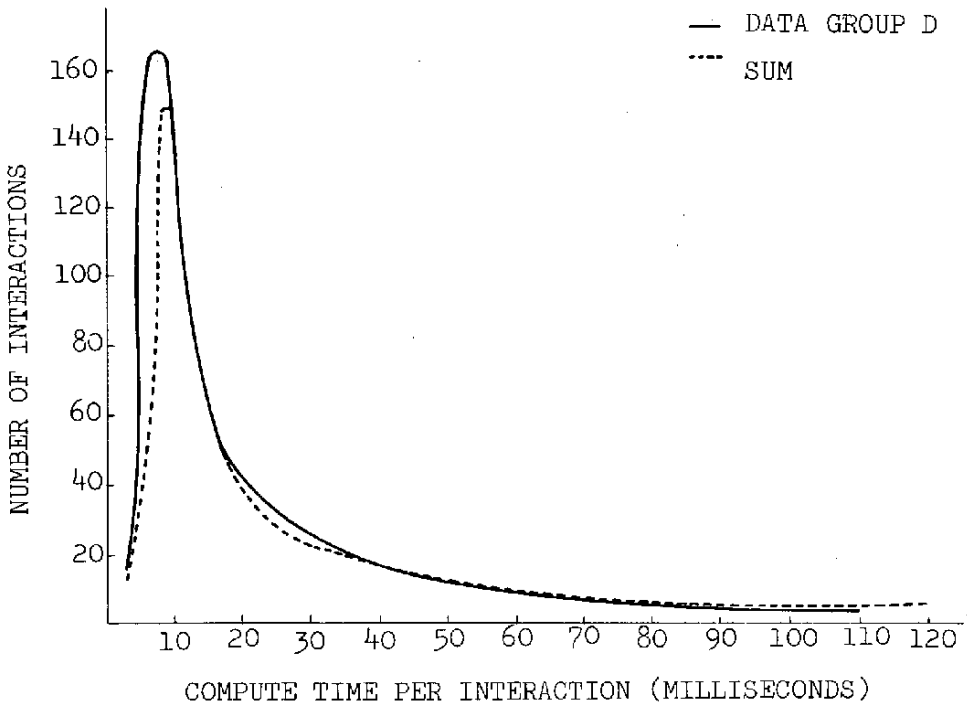


図4－3　Compute timeの分布

③ ページ・フォールト

1つのインタラクション中で発生するページ・フォールトは各プログラム構造に依存するだけでなく，オペレーティング・システムのタスク・スケジューリングとかページ・リプレースメント（page-replacement）アルゴリズムと言った機能に依存する部分も多い。そこで，なるべくオペレーティング・システムに依らないページ・フォールトだけを取り出して考えている。この様なページ・フォールトのことをunique page faultと言っている。しかし実際には，インタラクションを基準に比較している意味もあって，ページ・リプレースメント・アルゴリズムに依存しがちになっている。インタラクションと言うのは前にも述べたように，続けて出されるターミナルへのＲＥＡＤ要求の間で行われる処理（activity）をさしている。この間，ターミナルに対するWRITEが含まれているのが普通であるが，あるタスクからWRITE要求が出されると，そのタスクはインアクティブ（inactive）状態にされ，他のタスクのページをメモリーに読み込んでくる事が可能な状態になる。従ってWRITE要求を出したタスクのページは2次記憶に追い出され，変って他のタスクがメモリーを占有することになる。ところが，WRITEが完了すると再び，そのタスクはアクティブ（active）に成るので再びページ・リプレースが行われる。その為にインタラクションごとのページ・フォールト数とユニーク・ページ・フォールトの比が大きくなってしまう。

次の表4－3と表4－4はそれぞれインタラクションごとのページ・フォールトとユニーク・ページ・フォールトのデータを示しており，図4－4，図4－5，図4－6はその分布をグラフに書いたものである。ユニーク・ページ・フォールトの分布でピークが2つある事に関しては，個々のプログラムの性格の相異が原因になっているのではないかと述べている。

表4－3 PAGE FAULTS PER INTERACTION

	OBSERVED DATA				NORMALIZED DATA				
	A	B	C	D	A	B	C	D	SUM
#INTER-ACTIONS	970	1304	1367	1436	1293	1293	1293	1293	1293
MEAN	35.5	29.6	31.5	37.9	19.4	16.3	17.7	18.0	17.9
MEDIAN	17.5	13.6	14.8	14.7	15.2	12.7	13.4	12.4	13.2
MODE	8.5 (5.0%)	3.5 (5.8%)	6.5 (6.1%)	4.5 (6.8%)	8.5 (5.5%)	3.5 (6.4%)	6.5 (6.8%)	4.5 (7.6%)	4.5 (5.6%)
STANDARD DEVIATION	121.1	80.1	72.9	110.5	16.1	13.6	15.2	17.1	15.6

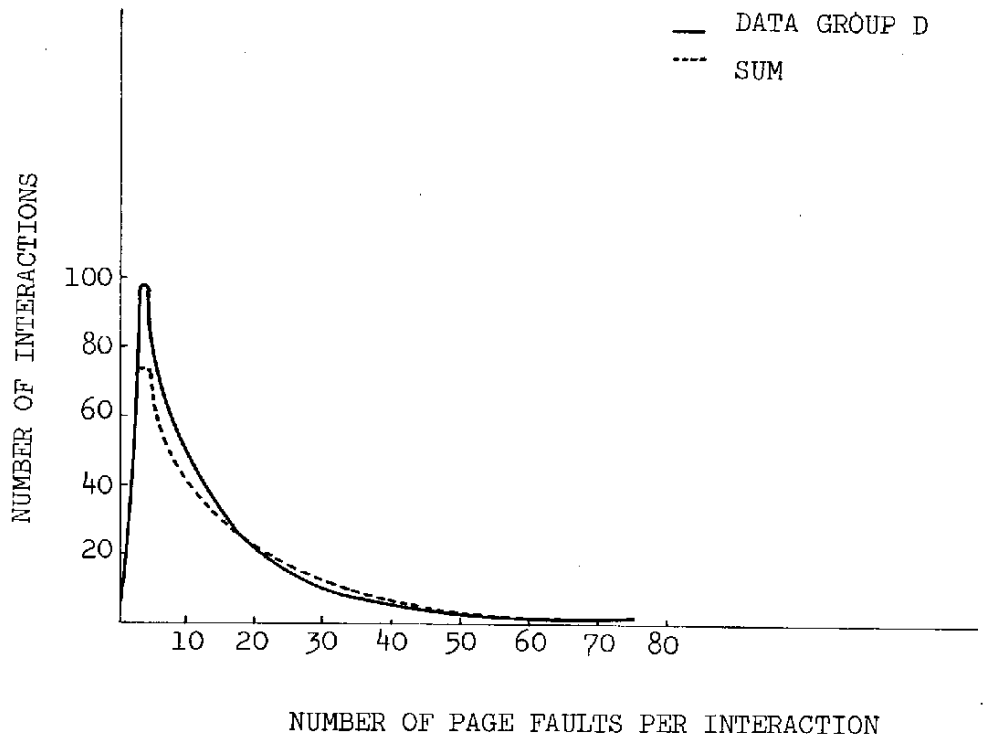


図4－4 ページ・フォールト回数の分布

表4－4　UNIQUE PAGE FAULTS PER INTERACTION

	OBSERVED DATA				NORMALIZED DATA				
	A	B	C	D	A	B	C	D	SUM
#INTER-ACTIONS	970	1304	1367	1436	1293	1293	1293	1293	1293
MEAN	15.8	16.9	18.2	17.5	11.7	12.5	12.8	12.2	12.3
MEDIAN	11.9	11.5	12.0	11.7	11.5	11.0	11.3	10.7	11.2
MODE	8.5 (6.1%)	3.5 (6.0%)	6.5 (6.0%)	4.5 (8.3%)	8.5 (6.8%)	3.5 (6.7%)	6.5 (6.7%)	4.5 (9.2%)	4.5 (6.8%)
STANDARD DEVIATION	15.4	16.6	22.3	22.7	7.5	9.3	9.6	8.8	8.8

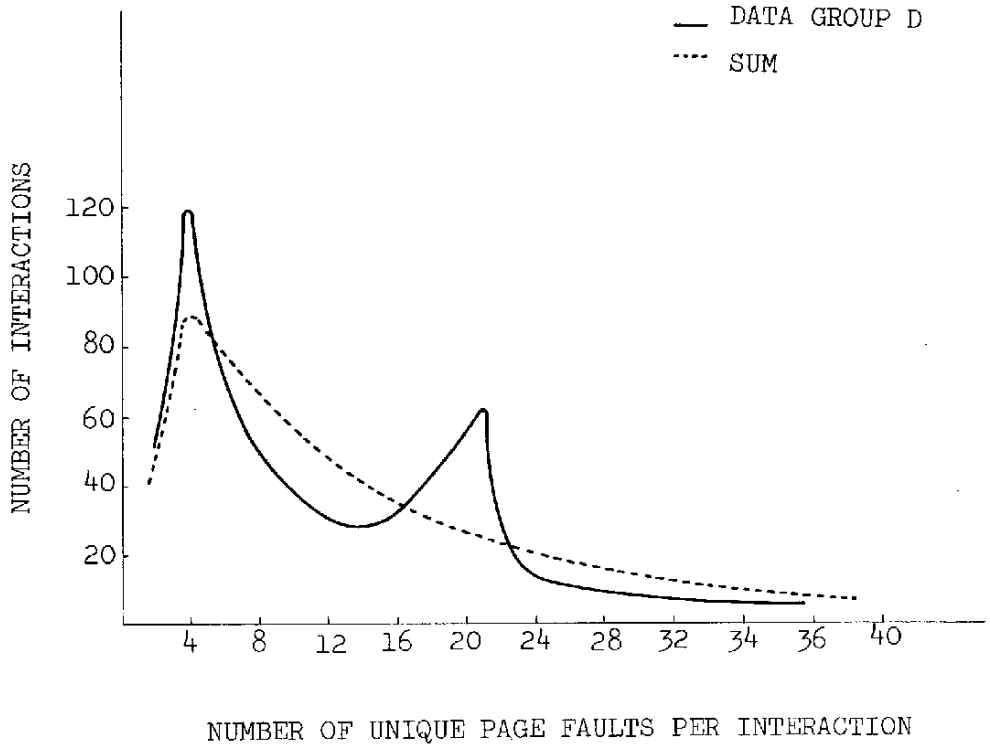


図4－5　ユニーク・ページ・フォールト回数の分布

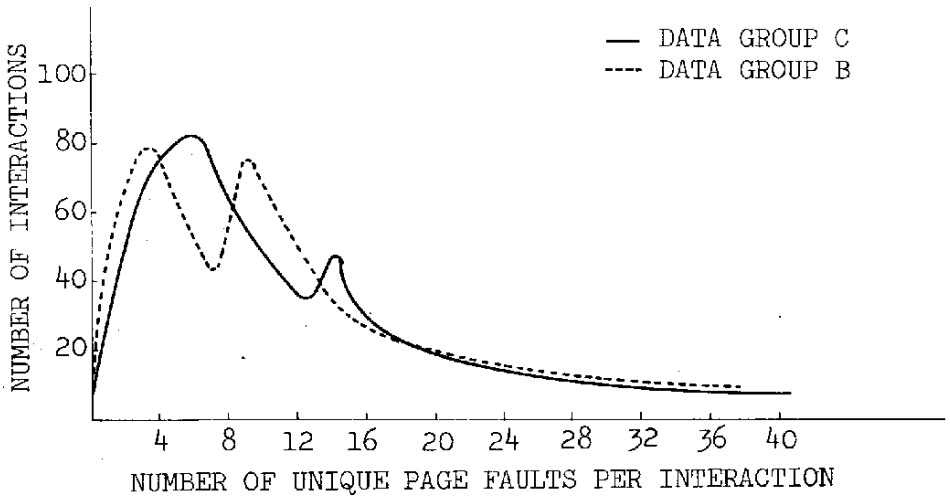


図４－６　ユニーク・ページ・フォールト回数の分布

④　ページ・フォールト間のコンピュート・タイム

次の表４－５と図４－７は続けて起るページ・フォールト間でのコンピュート・タイムに関するデータと分布を示したものである。

表４－５　COMPUTE TIME BETWEEN PAGE FAULTS(MILLISECONDS)

	OBSERVED DATA				NORMALIZED DATA				
	A	B	C	D	A	B	C	D	SUM
#PAGE FAULTS	40740	42750	48891	59940	53890	53890	53890	53890	53890
MEAN	6.7	9.9	10.6	8.5	1.1	1.4	1.5	1.2	1.3
MEDIAN	0.5	0.6	0.6	0.5	0.5	0.6	0.6	0.5	0.5
MODE	0.2 (24.6%)	0.2 (18.7%)	0.2 (19.3%)	0.2 (23.4%)	0.2 (27.4%)	0.2 (20.9%)	0.2 (21.5%)	0.2 (26.2%)	0.2 (24.0%)
STANDARD DEVIATION	65.1	88.2	86.7	80.8	1.5	1.9	2.0	1.6	1.8

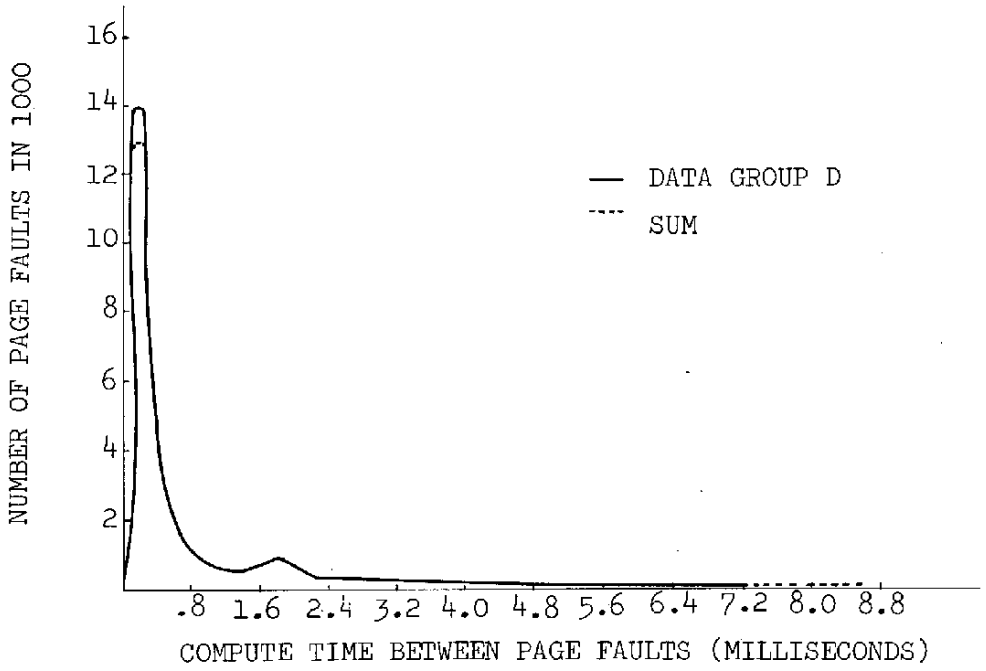


図4－7　ページ・フォールト間のコンピュート・タイム分布

⑤　I／O 分 布

表4－6はインタラクションごとに出される周辺入出力装置へのアクセス回数の統計量でその分布が図4－8に示してある。

表4－6　インタラクションごとのペリフェラルI／O回数

	A	B	C	D
# INTERACTIONS	1051	1493	1514	1500
MEAN	10.9	13.3	15.4	9.9
STANDARD DEVIATION	83.1	54.0	94.3	56.0
PERCENT INTERACTIONS WITH NO I/O	61.5	58.2	60.9	73.1
80th PERCENTILE	11	9	11	6
90th PERCENTILE	20	28	26	15
95th PERCENTILE	38	48	53	35

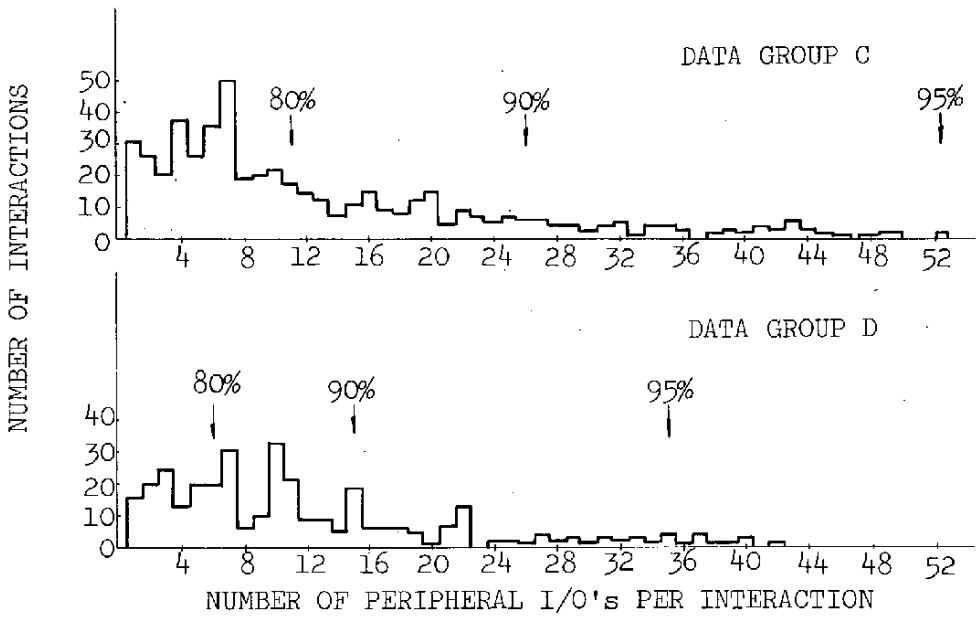


図4－8　ペリフェラル I/O　回数の分布

4.2.2　Pseudo－programsを使ったＴＳＳ評価

ＲＣＡではTime Sharing Operating System（TSOS）を開発中にパフォーマンスの評価，解析を行った例を報告している。この評価では，Pseudo－programを用い，ワークロードを自由にコントロール出来る様な状況を作り出している点が興味深い。プロセッサーの利用率やページング用ドラムの利用率に対するリスポンス・タイムの関係，スケジューリングに「カンタム・タイム・スライシング」方式を用いた効果，スラッシング（Thrashing）の問題等について解析を行っている。

A.　Pseudo－program

普通の端末ユーザの振舞いをなるべく忠実に反映する様に作られたプログラムで，各種リソースの要求も適当な配分で行う様に成っている。このプログラムの動作を大まかに示したのが次の図4－9である。

図4－9で見る通り同じ様な動作を繰り返すプログラムであるが，1msecＣＰＵを使い終ると，次にまたＣＰＵを使うか，他のリソースを使うかが，パラメータで与えられる様に成っている。

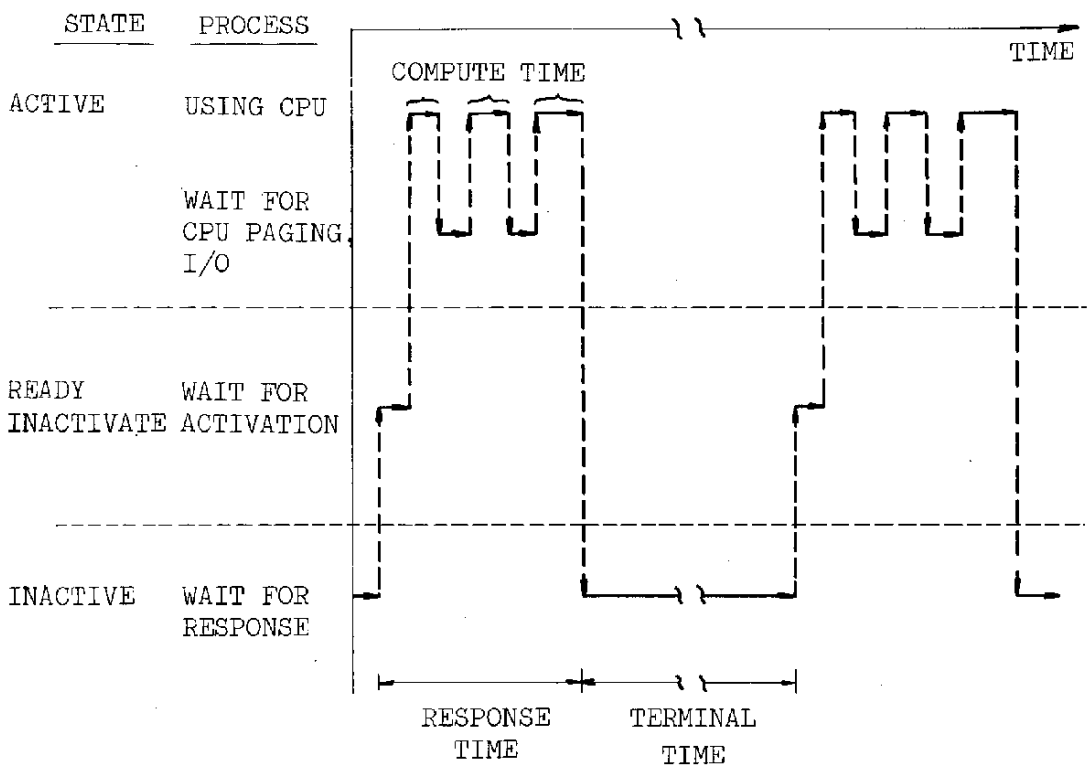


図4－9　Pseudo－programの動作

B．測定結果と解析

Pseudo－programで適当なロード状況を作り出し実際に測定した結果が述べられている。しかし，個々のデータ，例えばＣＰＵの利用率ひとつにしても，どの様な方法で求めたのか（ソフトウエア・モニターを使ったとかハードウエア・モニターを作ったと言う）が説明されていない。

次の表4－7は図4－10，4－11に示したタスク数とリスポンス・タイム，ＣＰＵ利用率とリスポンス・タイムの関係を求める際に使ったPseudo－programの明細を表わしている。（メモリーサイズの割にはＣＰＵを余計使うジョブが多い。）

同じようにして，ページング用に使っているドラムの利用率とリスポンス・タイムの関係を求めている。表4－8はこの時使ったジョブ・ロードの明細がしめしているが，ページング・リミットに成る様に選ばれている。図4－12はその結果をグラフに示したものである。

表４－７　測定に使ったジョブ・ロード

LOAD* LABEL	COMPUTE TIME (msecs)	TERMINAL TIME (secs)	READ PAGES	WRITE PAGES	I/O'S/ CYCLE	VARY PAGES
□	500	15	1	9	NONE	NONE
△	300	15	1	9	NONE	NONE
○	100	15	5 (av)	6 (av)	2	± 1/4
▲	300	15	5 (av)	6 (av)	2	± 1/4
●	A mix of tasks, consisting of multiple sets in which each set contains the following tasks:					
	100	15	2	2	NONE	± 1/4
	300	15	3	3	NONE	± 1/4
	500	15	4	4	NONE	± 1/4
	700	15	5	5	NONE	± 1/4

*同じロードラベルでＣＰＵ利用率の増減はタスク数を変えて調整している。

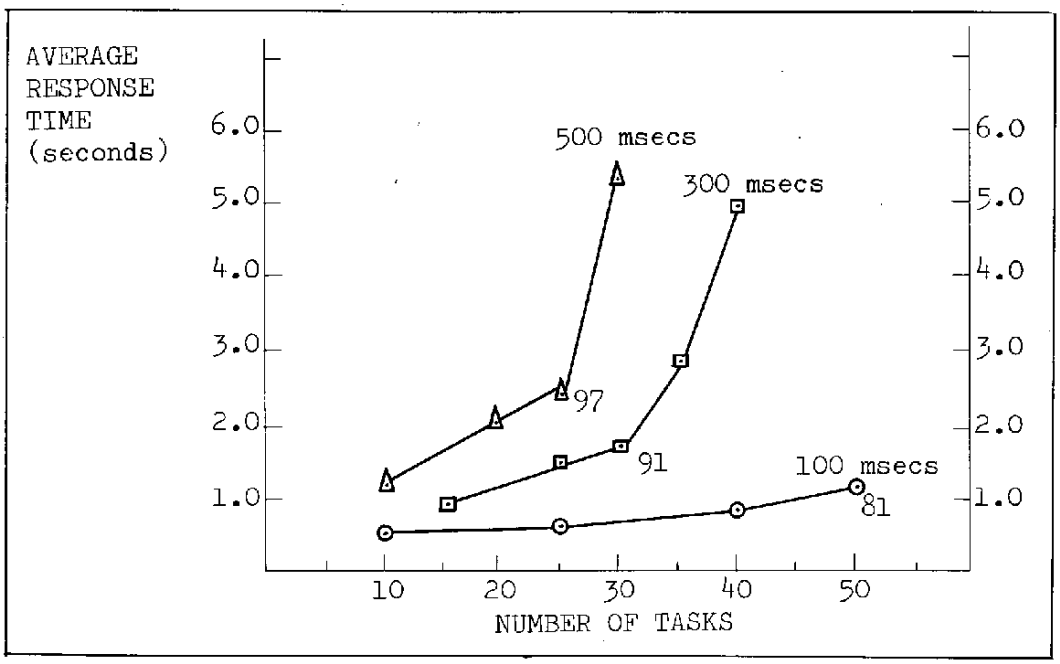


図４－10　タスク数とリスポンス・タイム

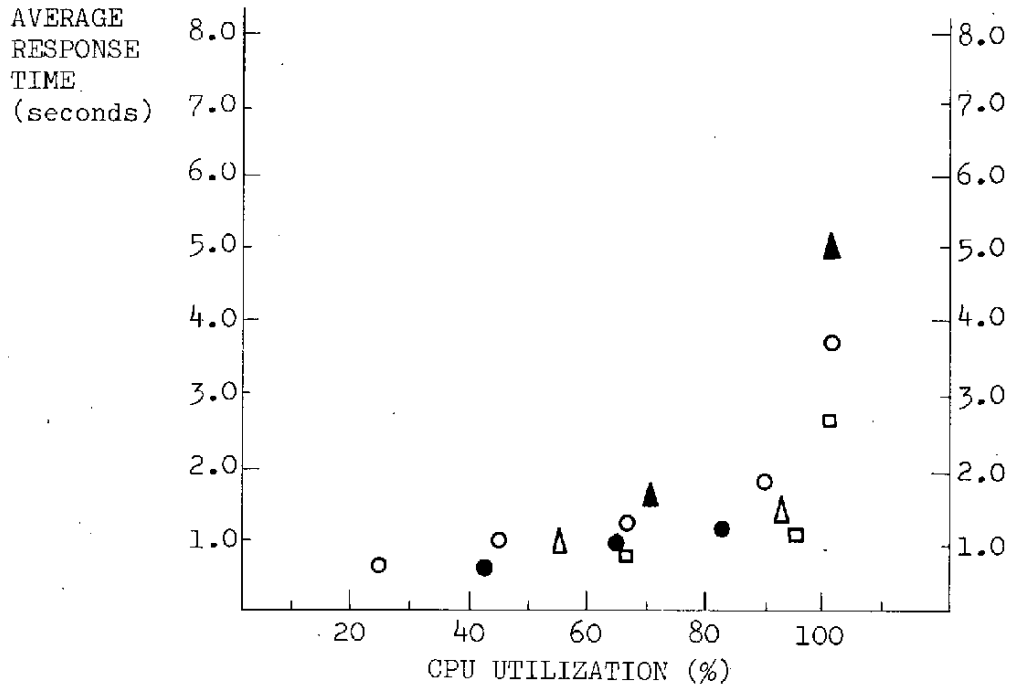


図４－11　ＣＰＵ利用率とリスポンス・タイム

表４－８　測定に使ったジョブ・ロード

LOAD* LABEL	COMPUTE TIME (msecs)	TERMINAL TIME (secs)	READ PAGES	WRITE PAGES	I/O's CYCLE	VARY PAGES
○	100	15	1	9	NONE	NONE
●	100	15	5 (av)	6 (av)	2	± 1/4
⊗	A mix of tasks, consisting of multiple sets in which each set contains the following tasks:					
	20	15	1 (av)	2 (av)	NONE	± 1/4
	30	15	2 (av)	3 (av)	NONE	± 1/2
	50	15	3 (av)	4 (av)	NONE	± 1/2
	100	15	4 (av)	5 (av)	NONE	± 1/2
	150	15	5 (av)	5 (av)	NONE	± 1/2
	150	15	5 (av)	6 (av)	NONE	± 1/2
	200	15	6 (av)	7 (av)	NONE	± 1/2
	250	15	7 (av)	8 (av)	NONE	± 1/2
⊠	Same as above, except VARY PAGES = ± 1/4					
⊕	Same as above, except VARY PAGES = ± 1/4 and TERMINAL TIME = 10 seconds					

*同じロードラベルでドラム利用率の増減はタスク数を変えて調整している。

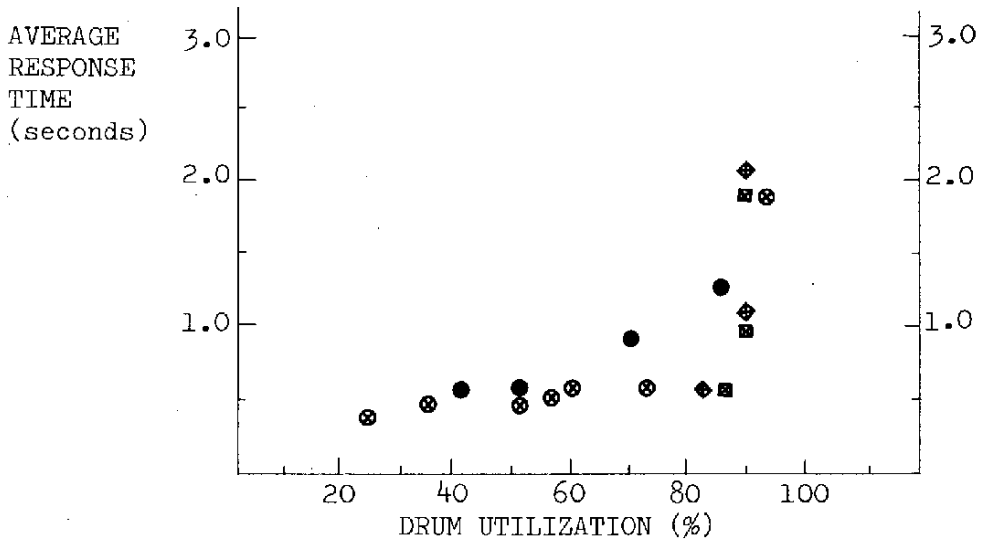


図4－12　ドラム利用率と平均リスポンス・タイムの関係

これら2つの結果を見るとＴＳＯＳのスケジューリングはメモリー・スペースとプロセッサーについて比較的効率良い使い方をしている事がわかると述べている。

① タイム・スライスイング

図4－13はTSOSのversion BとDについて，プロセッサー利用率と平均リスポンス・タイムの比較をしたものである。

この測定はＴＳＯＳの場合カンタムタイム・スライシング方式がどの程度効果的であるかを調べたものである。

version Bのスケジューリングは2秒間の個定タイム・スライス方式でプロセッサー・キューのサービス方式はＦＣＦＳ（First Come First Service ）方式で行っている。

これに対しversion Dでは1秒のタイム・スライスを，さらに5等分し，200 msecのカンタムタイム単位でRound－Robin方式のサービスを行っている。結果のグラフから明らかな様にversion Bではプロセッサー利用率が70～80％程度でリスポンス・タイムが急に上昇しているが，version Dでは90％ぐらいまでゆるやかな上昇を続けている。この様な事からもカンタムタイム・サービス方式が効果的だと述べている。

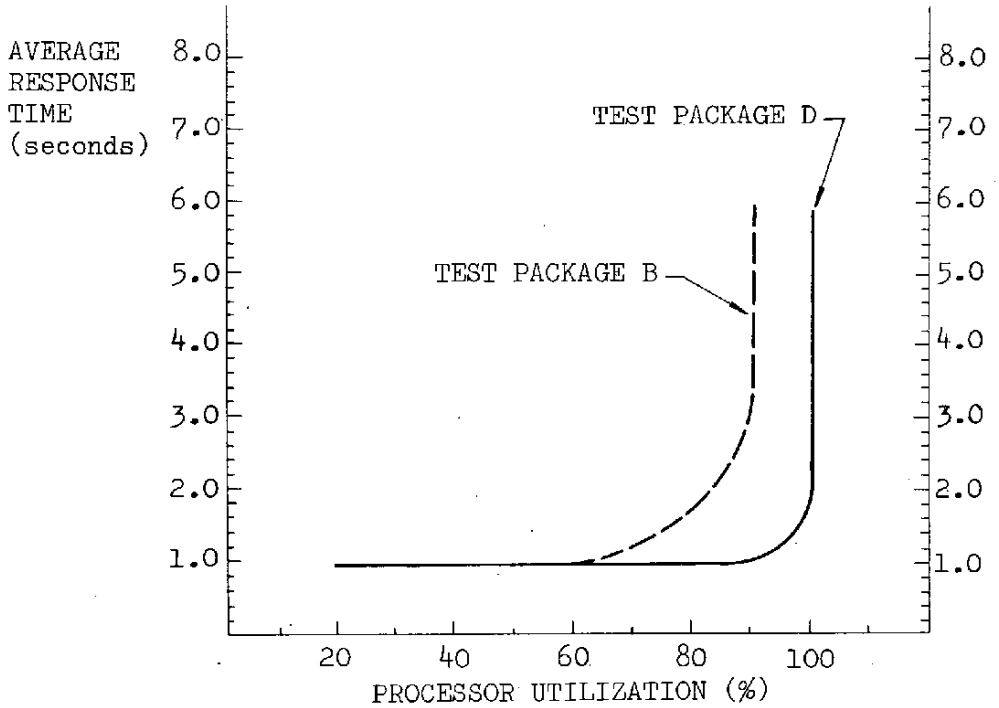


図4－13　タイム・スライシング効果の比較

② Thrashing

Thrashingの問題はDenningが指摘している通り，過度のページトラフィックに依ってシステムover headが急に増し，パフォーマンスを低下させる現象である。Denningはこの現象について簡単な解析を行い次の様な説明を加えている。

平均タスク数をNとし，その平均ページ・サイズをSとする時，メイン・メモリーに入り得る最大ページ数Mとの間に

$$N \cdot S < M$$

の関係が成り立つうちはこの現象は起らず，あらたに，タスクが発生したりして

$$(N+1) \cdot S \geq M$$

となると，とたんにThrashingが発生する。

次の図4－14はThrashingが起った時，リスポンス・タイムが急激に低下している様子を示している。

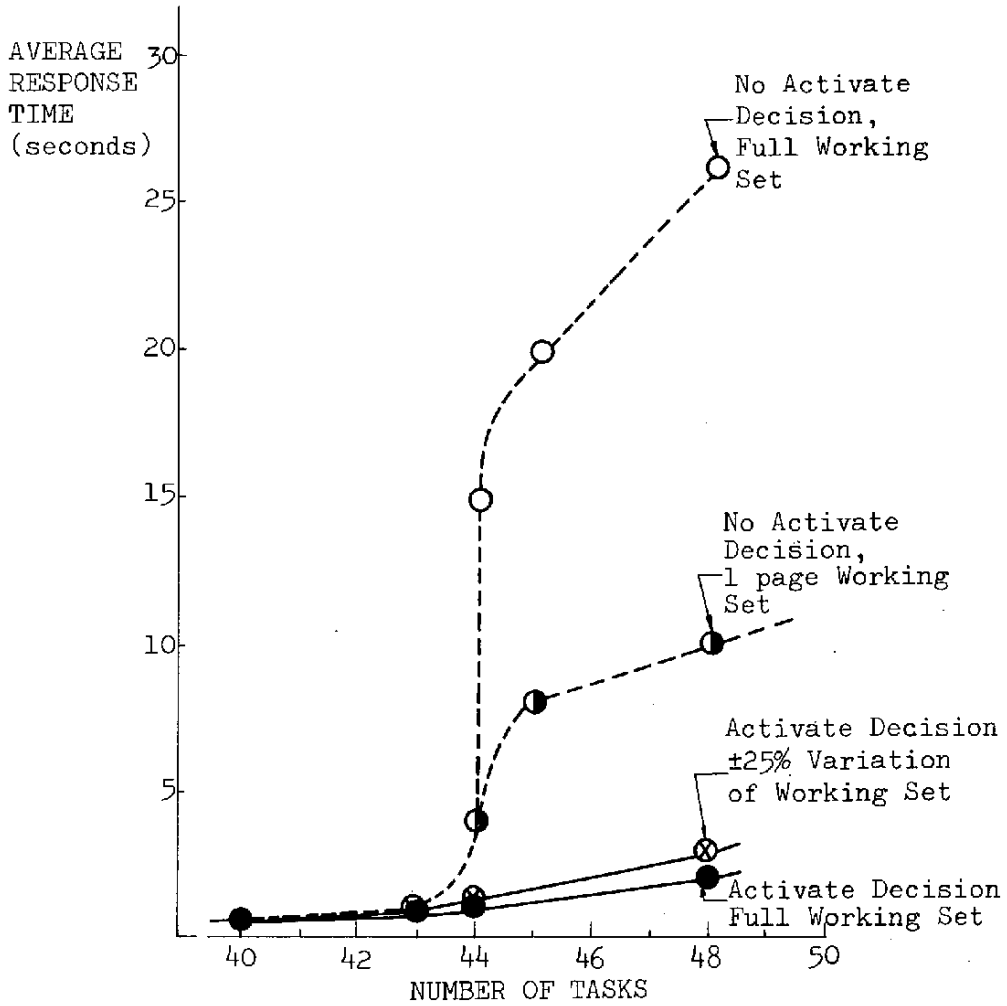


図4－14　Thrashingの様子

4.2.3　タイム・シェアリング・システムのシミュレーション

RCAのJ. Winograd らは，ヴァーチャル・メモリー（Virtual memory），タイム・シェアード（Time-shared），デマンド・ページ（Demand page），オペレーティング・システムのシミュレーション・モデルを開発し，このモデルを使ってシミュレーション実験を行い，結果の一部を報告している。SIM／61と呼ばれるこのモデルはオペレーティング・システムの割込み処理，タスク・スケジューリング，I／Oスケジューリング，ページングと言った機能も組み込まれており，かなりディテール・モデルとなっている。プ

ログラム自身はイベント・タイプの汎用シミュレーション言語SIMSCRIPT 1.5を使ってコーディング(約3500 ステップ)している。シミュレーション・モデルのあまり細かい事については触れていないが，ワークロードとしてオペレーティング・システムのもとで実行されるいろいろなタスクのセットを与えている事が述べられている。個々のタスク特性を表わすデータとして，次の8項目を与えている。

① タスク・タイプ	バッチ・タスク，インタラクティブ・タスク，コミュニケーション・タスクの別を示す。
② ワーキングセット・サイズ	ワーキングセットに含まれるタスク・ページの数
③ コンピュート・タイム	各インタラクションで必要なサービス時間
④ スインク・タイム(Think Time)	インタラクション間の時間でタイプイン，タイプアウト時間も含んだもの。
⑤ I／Oインターバル	disk I／O の要求間隔
⑥ Exec SVC インターバル	SVC(Super Visor Call)の発生間隔
⑦ Shared page の参照間隔	Shared page を参照する時間間隔
⑧ ヴァーチャル・メモリー・サイズ	タスクが使用するヴァーチャル・メモリーのページ数

以上のパラメータは①と⑧以外，全て指定された平均値と標準偏差を持つ正規分布で与えている。

実際のシミュレーションでは，Fine らに依って実測されたデータをもとにして，これらのパラメータを与えた事が述べられている。

A. シミュレーション実験

シミュレーション実験は，①ページング・バウンドに於けるシステム・パフォーマンスの問題，②バッチ・タスクとインタラクティブ・タスク間のス

ケジューリング方法に関する検討，の２点について行いその結果をまとめている。

① Backing Store の研究

この実験では，システムをページ・バウンド状況に置いて，メイン・メモリー・サイズ，ページング・レイト（Paging rate）容量を変更した時パフォーマンスに及ぼす影響を調べている。

実際にはメイン・メモリー・サイズとページング用デバイスの構成を変更して実験しており，その時対象としたページング用デバイスの性能は次の表に示されている。

表４－９　ページング用デバイスの性能

DEVICE CHARACTERISTICS	RCA 8507 DRUM	RCA 8580 DISC	RCA 8590 DISC	HYPO-THETICAL IAM
TRANSFER RATE (KB)	333	800	312	900
ROTATION TIME (MS)	17.2	16.7	25.0	---
SEEK TIME - TRACK TO TRACK (MS)	---	10	25	---
SEEK TIME - AVERAGE (MS)*	---	30	75	---
SEEK TIME - MAXIMUM (MS)*	---	55	135	---

* デスクは全面が使用されている。

ロードとして与えたタスクは64のインタラクティブ・タスクと２つのコミュニケーション・タスク，２つのバッチ・タスクで十分にページング・デマンドの状況が発生する様にパラメータが選ばれている。実際に与えた各タスクのパラメータを示したのが次の表である。

この様な状況のもとでシミュレーションを行った結果が表４－11にまとめられている。また，図４－15 ～ ４－17 はページング・レイトを変数として，平均リスポンス・タイム，インタラクション数，ユーザＣＰＵ使用率との関係をグラフにしたものである。シミュレーションはシミュレーション・クロックで９分間行ったが，この処理に10分間必要だったそうである。

表4－10 LOAD FOR BACKING STORE STUDY

TASK TYPE*	# OF TASKS	WORKING SET SIZE (PAGES)		COMPUTE TIME (MS)		THINK TIME (SMC)		I/O INTERVAL (MS)		EXEC SVC INTERVAL (MS)		SHARED REF INTERVAL (MS)		TOTAL VIRT MEMORY SIZE (PAGES)
1	9	10	2	25	5	15	3	20.0	5.0	2.0	0.5	3.0	0.7	20
2	9	10	2	20	5	10	2	5.0	1.0	3.0	0.7	10.0	2.5	20
3	5	20	5	300	10	30	5	0	0	5.0	1.0	0	0	40
4	9	15	3	25	5	20	5	2.0	0.5	1.0	0.2	0	0	30
5	9	10	2	10	2	5	2	5.0	1.0	1.5	0.3	0	0	20
6	9	10	2	25	5	25	5	20.0	5.0	2.0	0.5	3.0	0.7	20
7	9	10	2	20	5	20	5	5.0	1.0	3.0	0.7	10.0	2.5	20
8	5	63	10	50	5	30	5	25.0	5.0	5.0	1.0	0	0	126
9	2	20	5	500	50	0	0	10.0	2.5	3.0	0.7	0	0	40
10	2	10	2	10	2	5	2	5.0	1.0	1.5	0.3	0	0	20

PARAMETERS ARE DRAWN FROM A NORMAL DISTRIBUTION WITH MEAN . STANDARD DEVIATION

*1 INTERACTIVE TASK, BASIC USER, SHORT THINK TIME
2 INTERACTIVE TASK, FILE EDIT USER, SHORT THINK TIME
3 INTERACTIVE TASK, COMPUTE BOUND EXECUTION
4 INTERACTIVE TASK, I/O BOUND EXECUTION
5 INTERACTIVE TASK, COMMUNICATIONS - LIKE BEHAVIOR
6 INTERACTIVE TASK, BASIC USER, LONG THINK TIME
7 INTERACTIVE TASK, FILE EDIT USER, LONG THINK TIME
8 INTERACTIVE TASK, PAGING BOUND EXECUTION
9 BATCH TASK
10 COMMUNICATIONS TASK

TOTAL OF 64 INTERACTIVE TASKS, 2 BATCH TASKS, 2 COMMUNICATIONS TASKS.

表4－11 PERFORMANCE RESULTS FOR BACKING STORE STUDY

	INTERACTIVE TASK AVG. RESPONSE TIME (SEC)		PAGING RATE (PAGES/ SEC)		USER I/O RATE (2048 BYTE DISC IOs/SEC)		USER		CPU (7) OVERHEAD		IDLE	
PAGING CONFIGURATION	1/2 MB	1 MB	1/2 MB	1 MB	1/2 MB	1 MB	1/2 MB	1 MB	1/2 MB	1 MB	1/2 MB	1 MB
2 RCA 8580 DISCS	48.1 514	33.0 667	36	40	3.4	4.3	3.2	4.0	10.5	11.6	86.3	84.4
1 RCA 8567 DRUM	21.8 883	17.5 1003	58	58	5.4	6.0	5.0	5.6	13.8	14.3	81.2	80.1
2 RCA 8567 DRUMS	7.8 1487	3.9 2051	96	114	8.4	12.0	8.0	15.7	21.4	26.5	70.6	57.8
DIAM	3.1 2157	1.3 2519	170	145	11.8	20.4	16.8	28.4	41.5	40.7	41.7	30.9

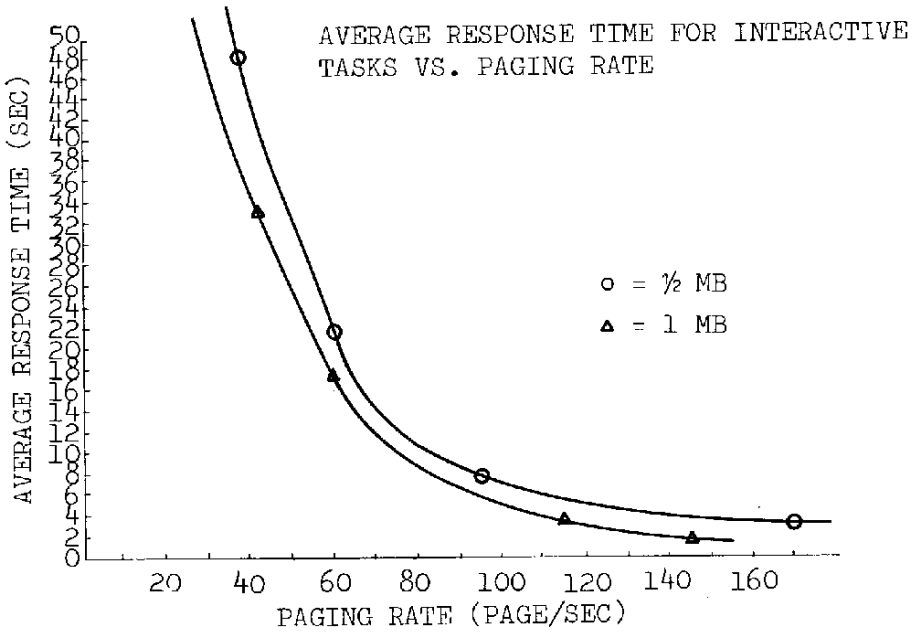


図4－15　ページング・レイトと平均リスポンス・タイムの関係

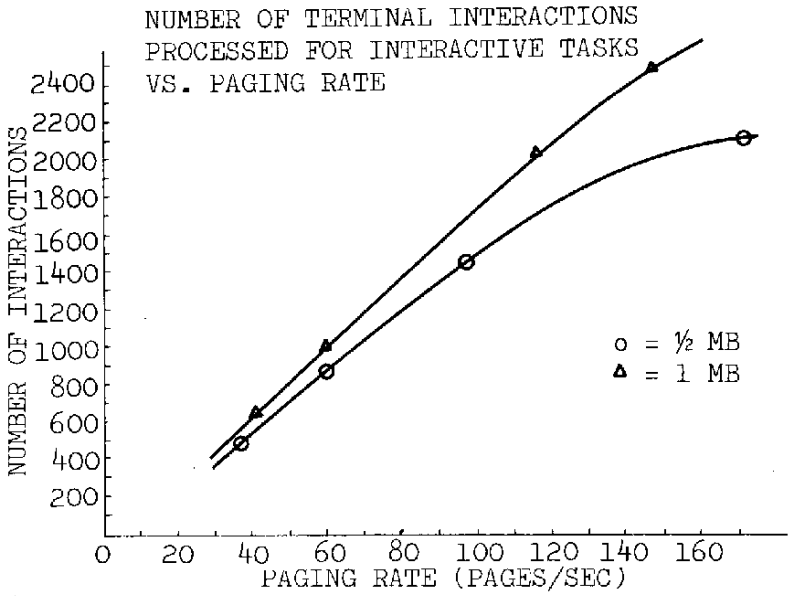


図4－16　ページング・レイトとインタラクションの関係

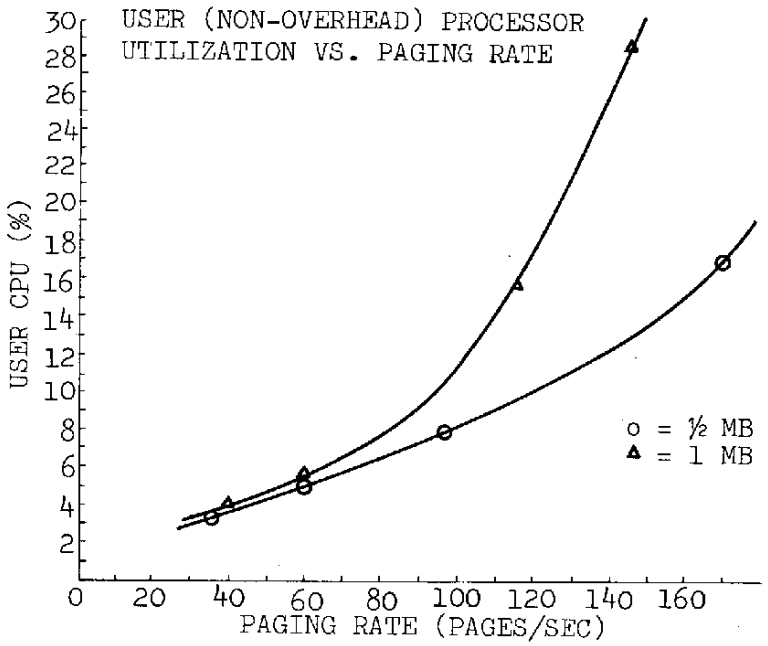


図4－17　ページング・レイトとＣＰＵ利用率

この結果については，幾つかのコメントが与えられている。例えばページング・レイトが増すとパフォーマンスが向上している事は，常識的に考えれば奇妙である。この事については，シミュレーションが通常のジョブ・ロード下で行われたものでなく，あくまで極度にページ・バウンドなジョブ・ロード下で行われており，ページ・レイトが上った事はページ・トラフィックがスムースに流れる様に成り，その為パフォーマンスが上っていると述べてある。

B.　スケジューリング・アルゴリズムの変更

①の実験で用いたタスク・アクティベイター（ task activater ）はインタラクティブ・タスクのリスポンスを良くする様な設計がされていて，バッチ・タスクに対しては，インタラクティブ・タスクのキューが空の時だけアクティブにする様なアルゴリズムと成っている。この様なスケジューリングなら適当な配分でバッチ・タスクへコントロールが渡ると思われていたが，実際には，極度なロード下ではバッチ・タスクには全然コントロールが渡ら

ない事が判明した。そこでタスク・アクティベイターを一部修正して，インタラクティブ・タスクにN回コントロールを渡すと1回バッチ・タスクにコントロールを渡す様にした。このアルゴリズムのことをN to 1アルゴリズムと呼んでいる。Nの値をパラメータとして変えた時に，システム・パフォーマンスに及ぼす効果を調べる為にシミュレーションを行なっている。シミュレーション時のハードウエア構成，ジョブ・ロードの詳細は以下の通りである。

○ロード

インタラクティブ・タスク …… 20

バッチ・タスク …… 1

インタラクティブ・タスクの内訳は

BASICユーザ，Think timeは15秒 …… 6

BASICユーザ，Think timeは10秒 …… 6

ページ・バウンド・タスク …… 2

コンピュート・バウンド・タスク …… 2

○ハードウエア構成

RCA 3 プロセッサー（ fixed point add time : 8.88μs ）

メイン・メモリー（ $^1/_4$ MB，64ページ ）

RCA 8567 ドラム（ページング用 ）

RCA 8590 デスク（ユーザI/O用 ）

シミュレーション結果を表4－12，図4－18～4－20までに示してある。

表4－12 N TO 1 ALGORTTHM

N	00	10	7	5	3	1	0	BATCH TASK STAND-ALONE
AVERAGE RESPONSE TIME (SEC)	12.2	14.7	15.4	16.3	18.5	21.3	27.8	--
NUMBER OF TERMINAL INTER-ACTIONS PROCESSED	454	403	392	377	347	316	260	--
BATCH CPU 7	0	6.5	8.4	10.4	14.1	19.7	28.2	71.8
BATCH ELAPSED TIME	00	154	119	96	71	51	35	14

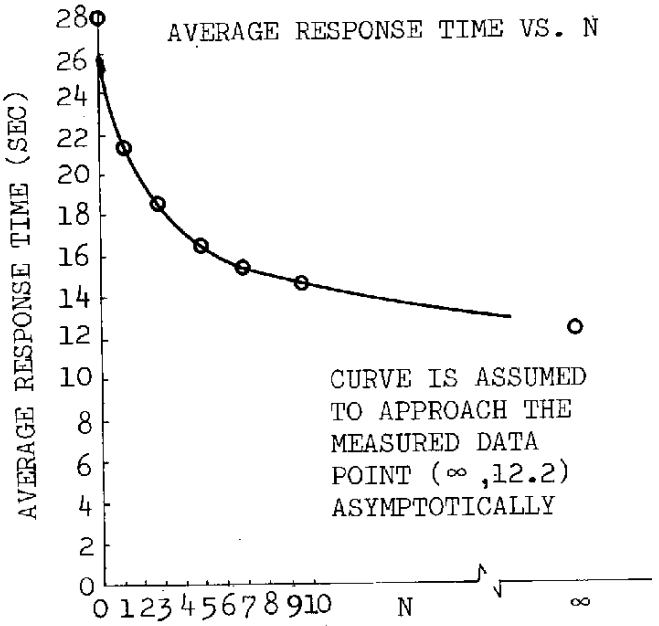


図4－18　Nと平均リスポンス・タイムの関係

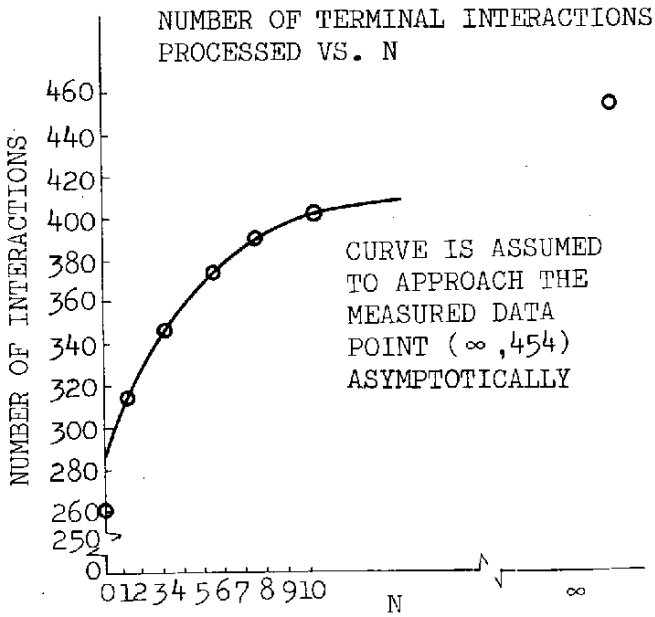


図4－19　Nとターミナル・インタラクションとの関係

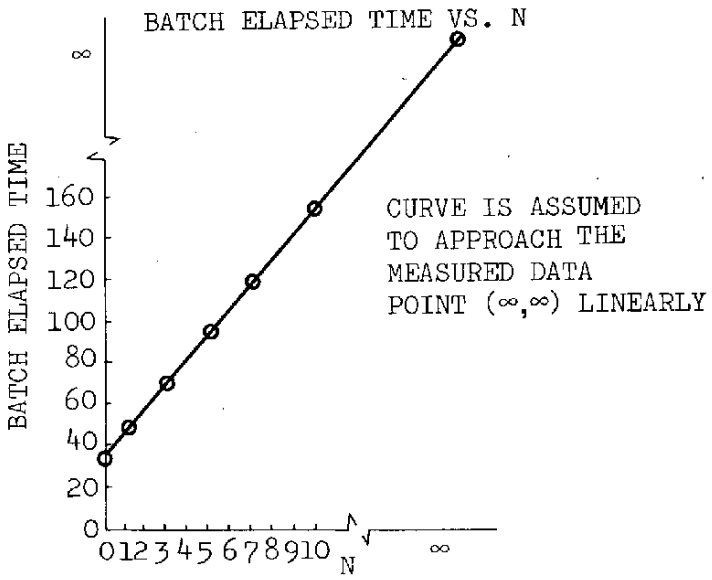


図4－20　Nとバッチ処理時間の関係

4.2.4　CDC 6600 UT－1システムの評価

テキサス大学では，大型コンピュータ・システムであるCDC6600（オペレーティング・システムはUT－1）のパフォーマンス評価を行った。同じ状況を繰り返し作る為にsynthetic job stream gnerator を使い，使用可能なリソースに制限を加えたり，リソース・アロケーション・アルゴリズムを変更したりして実験を行っている。実験中のシステム動作のデータはイベント・タイプのソフトウエア・モニタを使って記録し，そのデータをもとに，リソースの使用率，リソース使用にともなって出来る　行列パターンの解析を行っている。また，測定に使ったソフトウエア・モニタのオーバ・ヘッドについても述べられている。

A. 評価のねらい

この実験の目的とする所は，むしろ研究的なもので，次の3つが挙げられている。

(1) リソース自身の変化やリソース・スケジューリング・アルゴリズムの変化がシステム・パフォーマンスに及ぼす影響について調べる事。

(2) システム運用上の詳細な動作特性を調べる。

(3) イベント・タイプのソフトウエア・モニタが，システム・パフォーマンスを評価する上で，どれだけ有効な動具と成り得るかを調べる。

B. ソフトウエア・モニタについて

この実験に使用したソフトウエア・モニタはＥＤＳＭと呼ばれオペレーティング・システムＵＴ－１に手を加えて組込んだものである。一般に，ソフトウエア・モニタ（ソフトウエア・プルーブとも言われる）は，２つのタイプに分類され，１つはイベントが発生するごとに，その情報を記録するものでイベント・タイプと言い，他の１つは，適当な時間間隔で情報を取り出し記録するサンプリング・タイプのものである。ＥＤＳＭはこの内，前者に属するものである。ＥＤＳＭが記録する情報は，ＣＰＵサービスの要求，Ｉ／Ｏサービスの要求，ある種のモニタ機能の要求，コア・スペースの要求等に関するもので，さらに詳しくは次の表４－１３に示してある。

表４－１３　イベント・レコードの例

```
 I.  Requests by PP's
     1.  Request/Release Channel (n)
     2.  Request/Release Disk Half-track
     3.  Request/Release Storage
     4.  Request CP be assigned to Control Point (n)
     5.  Recall CP (Take control point out of ready status)

 II. Requests by User Programs
     1.  Perform I/O service
     2.  Recall CP (Take out of ready status)
     3.  End of Job-step

III. Activities of System Monitor
     1.  Assign CP to control point n
     2.  Assign PP to process user program request
     3.  Compact storage

 IV. Activities of Job Scheduler
     1.  Short job x at control point (n)
     2.  Roll-out control point (n)
     3.  Roll-in job x at control point (n)
```

これらのイベントに対応して，その種類を示す情報，発生時刻，その他の情報から成るレコードが磁気テープに書き込まれる。システムが持っているクロックの解像度はミリセコンド・レベルである。イベントの量は，通常のプロダクション処理中の測定で 4037 秒間に約 180万個が記録された。従って１秒間に約 4～500 のイベントが発生している事に成る。

このレポートではソフトウエア・モニタを組込んだことに依る，種々のオーバ・ヘッドの解析を行っている事が特徴的である。一般にソフトウエア・モニタはハードウエア・モニタと異り測定対象のシステム内に直接組込まれる意味もあって，オーバ・ヘッドが大きく，システム自身に与える影響も大であると言われている。一部には，その為に使いものにならないとする意見すらある。しかし，これはソフトウエア・モニタの組込み方にも色々と方法があり，オペレーティング・システム内でも非常にトラフィックの多い割込み処理機構などにプローブを挿入する場合特に問題が起るが，それほどトラフィックの多くない所では，あまり大きな影響は無い様である。この実験に用いたＥＤＳＭでは次の表に示す様なオーバ・ヘッドを持つ事が報告されている。

表4－14　ＥＤＳＭが使うリソース

Central Memory used for data buffers	1024 words (0.8% of 117,000 words available)	
Tape channel time	6 - 7% of elapsed time	
PP time (required to write full buffers on tape)	1% of total PP time	
Magnetic Tape Unit	1 of 6 available	
Test Results		
To run 20 test jobs	Without EDSM	With EDSM
Elapsed time (ET)	218 sec	218 sec
CP Busy (percentage of ET)	95.0%	95.5%
Total CP time	208,168 sec	208,880 sec
Total PP time	255,117 sec	256,629 sec

C. ジョブ・ジェネレータについて

この実験では，シンセティク・ジョブ・ジェネレータで作り出したジョブのセットをワークロードとして用いている。この様なジョブのセットは出来るだけ現実のロードに近い状態を再現してくれる事が望ましい。この報告で

は，実際にシステムが稼動している状態のデータをもとに，ジョブ・ジェネレータが作るジョブの性格を決めている。次に示す表4－15はこの様にして作ったジョブのセットがどれだけ現実のロードと一致しているかを調べ，その結果を示したものである。

表4－15 Comparison of Performance on Simulated and Actual Production Loads

	Actual Production			Simulated
	Run 1	Run 2	Run 3	Production
Job Completion Rate (jobs/hr)	492	456	399	456
CP Busy (Percentage of ET)	89%	82%	94%	89%
Disk Channels Busy				
System Channel	39%	50%	27%	50%
Scratch Channels (3)	45	59	32	76
Rates (per second)				
PP Functions	169	165	213	225
System Requests	45	30	36	69
Task Switches	142	146	147	141

シンセティク・ジョブについては，現実のロードを忠実に再現出来る必要はなく，むしろ，十分に近似されていれば良い事，及びロードの再現性を持つ点が有効である事，が指摘されている。

D. 実験とその結果

ＥＤＳＭ，ジョブ・ジェネレータを使って，スケジューリング方法や使用可能なリソースの状況によってシステム・パフォーマンスがどう変るかを調べる為に，次の6項目に渡る実験を行っている。

(1) 使用可能なスクラッチ・デスクの個数を変える。

(2) 使用可能なペリフェラル・プロセッサー（peripheral processor）の個数を変える。

(3) コントロール・ポイントの数を変える（マルチプログラミングの多重度を意味する）

(4) セントラル・メモリー・サイズを変える。

(5) デスク・ファイルのスペース・アロケーションを変える。

(6) ラウンドロビン・スケジューリングに於けるカンタム・サイズを変える。

これらの項目のうち使用可能なリソースの変更は，通常の機器構成に対して，一部を使えない状態にする事で行っている。

各項目に渡る実験の結果とそれに対して述べられているコメントについては以下に示す。一連の実験のまとめとして，この実験は統計的な手法として一変量解析法を用いているが，Grenander B Tsaoが多変量解析法がこの種の解析に有効である事を述べている事に対し著者らは，その場合にも，前もって一変量解析の手法を用いる必要性のある事を特に強張して結んでいる。

(1) スクラッチ・デスク数の変化

この実験は使用可能なスクラッチ・デスクの数を減らした時にスループット，リソース・マティリゼーション，待時間がどう変るかを調べたもので，その結果が表4－16，図4－21とである。

表4－16 Statistical Summary－Reduced Disk Availability

	Run 1*	Run 2	Run 3	Run 4
Elapsed Time (Seconds)	425	445	502	852
Job Completion Rate (jobs/hr)	424	404	358	205
Efficiency (H/ET x 100)				
System Channel	42%	66%	87%	79%
Scratch Channels (3)	105	77	59	0
CP	96	94	89	51
PP (8)	458	496	536	556
Percentage Wait (W/ET x 100)				
System Channel	22%	84%	194%	359%
Scratch Channel (3)	54	49	43	0
CP	32	271	262	148
PP (8)	13	10	42	122

*Run 1では3台のスクラッチチャネルを稼動した。
Run 2では2台の 〃
Run 3では1台の 〃
Run 4ではシステム用スクラッチチャネルだけを稼動した。

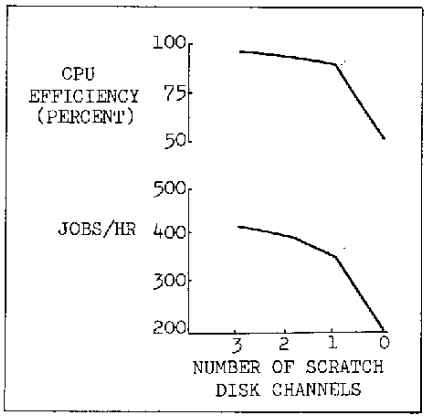


図4－21 RESULTS－REDUCED DISK AVAILABILITY

標準構成では3台のデスク（各々，別のチャネルに接続されている。）が，ユーザ用のスクラッチ・ファイルとして使われているが，通常の構成に対して，どの程度パフォーマンスが低下するかと言う形で扱われている。

(2) ペリフェラル・プロセッサー（Peripheral Processor）個数の変更

CDC6600ではペリフェラル・プロセッサーと呼ばれる，一種のサテライト・コンピュータが1台セントラル・プロセッサーの回りに配置されており，そのうちの2つはコントロール・プログラム用に，残りは主に入出力処理の為にプールされている。この実験では，入出力用にプールされているペリフェラル・プロセッサー（以下ＰＰと略記する）の台数を制限して，システム・パフォーマンスに及ぼす影響を調べている。その結果は次の表4－17と図4－22に示されているが，ＰＰ数が減少してもあまりパフォーマンスは低下していない。

これは，ＰＰのうち4～5台がリモート・ターミナル用やリモート・ステーションとの高速入出力制御用等に使われているが，今回の測定ではこの様なジョブを流していない為と説明している。

表4－17 Statistical Summary－Reduced PP Availability

	Run 1*	Run 2	Run 3	Run 4	Run 5
Elapsed Time (seconds)	420	434	434	448	453
Job Completion Rate (jobs/hr)	429	415	415	403	397
Efficiency (H/ET x 100)					
System Channel	53%	67%	61%	54%	52%
Scratch Channels (3)	90	86	83	85	77
CP	97	95	94	90	87
PP (8)	438	589	590	632	552
Percentage Wait (W/ET x 100)					
System Channel	46%	85%	64%	42%	30%
Scratch Channel (3)	20	20	17	25	15
CP	326	336	334	320	326
PP (8)	5	32	65	78	139

*Run 1で使用可能なＰＰは8台
Run 2で 〃 7台
Run 3で 〃 6台
Run 4で 〃 5台
Run 5で 〃 4台

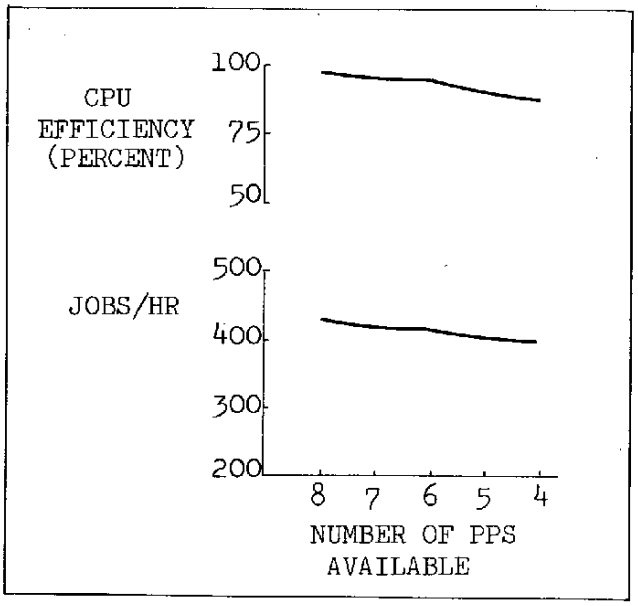


図4－22　RESULTS－REDUCED PP AVAILABILITY

(3)　マルチプログラミングの多重度を変更

マルチプログラミングの多重度を5から1まで変化させた時に，パフォーマンスに及ぼす影響を調べた結果が次の表4－18と図4－22である。

表4－18　Statistical Summary－Reduced Multi－programming

	Run 1*	Run 2	Run 3	Run 4	Run 5
Elapsed Time (seconds)	420	460	458	549	688
Job Completion Rate (jobs/hr)	429	392	393	328	262
Efficiency (H/ET x 100)					
System Channel	41%	68%	63%	43%	29%
Scratch Channels (3)	105	69	68	64	55
CP	96	92	91	76	60
PP (8)	423	437	395	254	242
Percentage Waiting (W/ET x 100)					
System Channel	20%	83%	48%	18%	3%
Scratch Channels (3)	33	8	7	9	0
CP	322	204	121	51	3
PP (8)	7	6	4	4	0

*Run 1の最大多重度は5
Run 2の　〃　4
Run 3の　〃　3
Run 4の　〃　2
Run 5の　〃　1

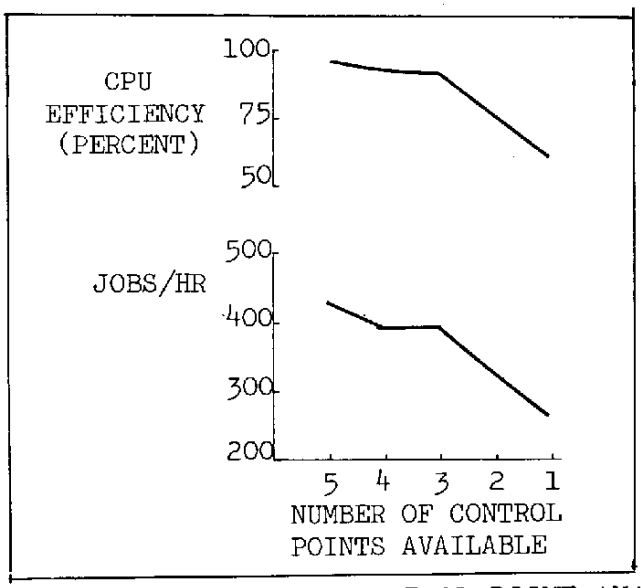


図4－23 RESULTS–REDUCED CONTROL POINT AVAILABILITY

(4) セントラル・メモリー・サイズを制限した場合

次の表4－19と図4－24はセントラル・メモリーのサイズを減らして行った時のパフォーマンス変化を調べたものであるが非常に興味深い結果が出ている。

表4－19 Statistical Summary–Reduced Availability of Central Memory

	Run 1*	Run 2	Run 3	Run 4	Run 5	Run 6
Elapsed Time (seconds)	423	432	444	443	446	450
Job Completion Rate (jobs/hr)	426	417	406	406	403	400
Efficiency (H/ET x 100)						
System Channel	55%	59%	59%	45%	48%	39%
Scratch Channels (3)	94	88	81	94	89	87
CP	97	96	93	93	93	91
PP's (8)	488	444	445	438	420	409
Percentage Wait (W/ET x 100)						
System Channel	44%	66%	64%	26%	39%	20%
Scratch Channels (3)	51	33	20	48	25	29
CP	290	271	302	279	262	211
PP's (8)	12	6	8	13	5	9

*Run 1で使用可能なCMは 128 K
Run 2で 〃 120 K
Run 3で 〃 112 K
Run 4で 〃 104 K
Run 5で 〃 96 K
Run 6で 〃 88 K

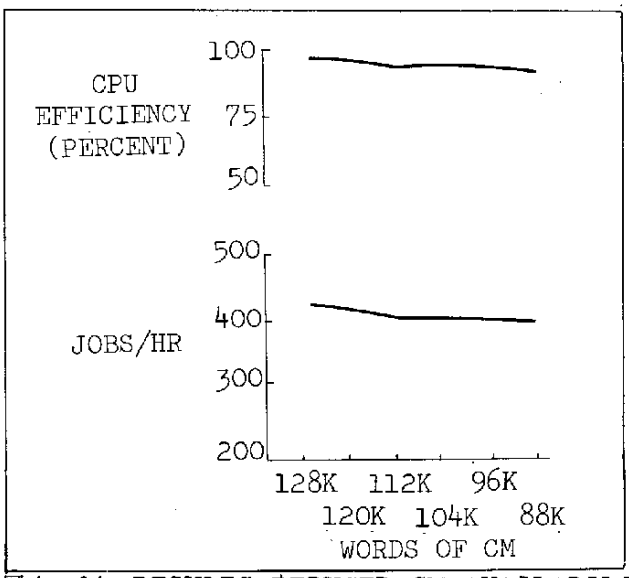


図4－24 RESULTS–REDUCED CM AVAILABILITY

最終的には40KWもメイン・メモリーを減らしたにもかかわらずＣＰＵ利用率，スループットともにほとんど変化していない。この事については，①スケジューリング・アルゴリズムの問題，②今回用いたジョブ・ミックスの平均コア・サイズが21KWで，マルチプログラミングの平均多重度も3程度なのでまだメモリーに余裕がある，などの点が説明されている。

(5) ディスク・スペースのアロケーション問題

ディスクのスペースにファイルをアロケートする時，出来ればファイル参照時のアーム移動がなるべく少なくなる様にしたい。この実験ではパーマネント・ファイルのアロケートを，①アーム移動を少なくする様な配置（これは実験的に決めた）と②極端に分散して配置し，アーム移動が頻繁に起る様にした場合でどの様にパフォーマンスが変るかを調べている。その結果かなり影響のある事が示されている。（表4－20，図4－25）

表4－20 Statistical Summary－Increased Disk Half－track Scattering

	Run 1*	Run 2
Elapsed Time (seconds)	425	470
Job Completion Rate (jobs/hr)	423	383
Efficiency (H/ET x 100)		
System Channel	56%	60%
Scratch Channels (3)	92	108
CP	96	91
PP's (8)	381	538
Percentage Wait (W/ET x 100)		
System Channel	46%	83%
Scratch Channels (3)	24	63
CP	306	237
PP's (8)	8	17

*Run 1 had normal disk-track assignment
Run 2 had scattered disk-track assignment

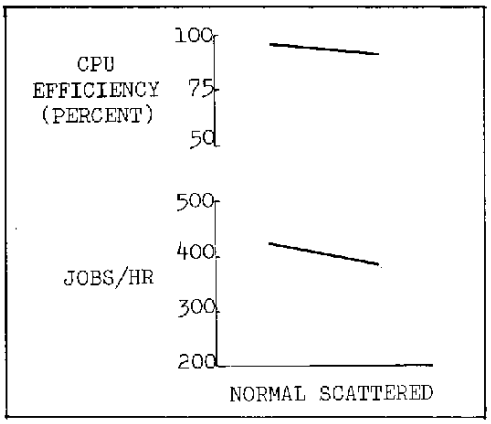


図4－25 RESULTS－ALTERED DISK TRACK ALLOCATION SCHEME

(6) ラウンド・ロビン（Round-Robin）スケジューリングのカンタム・サイズ

ＵＴ－１はラウンド・ロビン方式のスケジューリングを行っており，通常カンタム・サイズは８msec に成っている。これを２ｓｅｃ にした場合と比較し，カンタム・サイズがパフォーマンスに及ぼす影響を調べた実験で，その結果，Baskett が予測した通り，カンタム・サイズの小さい方が，Ｉ／Ｏリソース使用効率の良い事が認められたと述べている。次の表４－21 と図４－26 がその結果である。

表4－21　Statistical Summary—Alteration of CPU Scheduling Scheme

	Run 1*	Run 2
Elapsed Time (seconds)	421	453
Job Completion Rate (jobs/hr)	427	397
Efficiency (H/ET x 100)		
System Channel	49%	43%
Scratch Channels (3)	99	92
CP	97	93
PP's (8)	451	336
Percentage Wait (W/ET x 100)		
System Channel	42%	31%
Scratch Channels (3)	30	32
CP	307	284
PP's (8)	2	3

*Run 1のCPタイムスライスは8msec.
Run 2の　〃　2 sec.

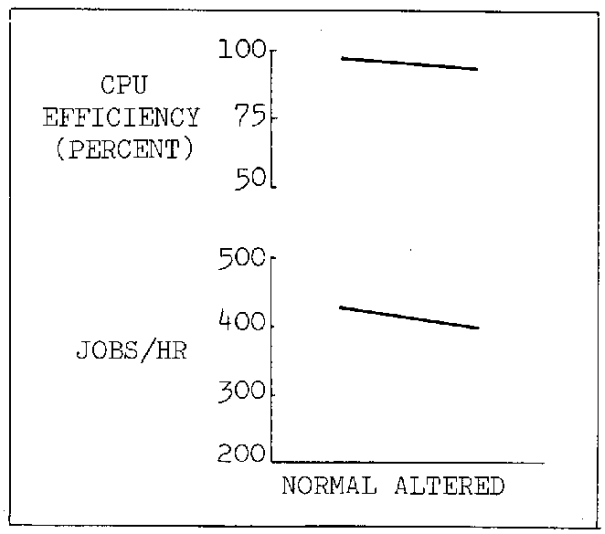


図4－26　RESULTS—ALTERED CP SCHEDULING SCHEME

4.3 解析モデル

計算機システムの解析モデルはほとんどが待行列理論に基づくものである。初期の，どちらかと言えば単純なモデルからマルチ・リソースまでを取扱ったネットワーク・モデルに到るまで，種々のモデルが研究されて来た。しかし，現在も尚，解析モデルへの期待とその反面，モデル自身の単純さからくる精度不足と言った不満も残されている様である。この節では，初期のモデルからネットワーク・キューイング・モデルに到るまでの研究を概観し，モデルの分類を行い，各モデルの特徴などをまとめてある。その内容は一番オーソドックスなカンタム・サービス方式モデル，ドラムやディスク・システムを扱った解析モデル，キャパシティー（Capacity）問題を取扱ったモデル，プライオリティーを与えたコンベンショナル・プライオリティー(Conventinal Priority)方式モデル，ネットワーク・キューイング・モデルから成っている。

4.3.1 カンタム・タイム・サービス方式（Quantum Controled Senvice Descipline）

サービスを受ける為にキュー（Queue）で待っているジョブのサービス順序をきめる方式は色々と考えられている。

例えばＦＣＦＳ（Fivst Come Fivst Service）方式は代表的な例である。これは，説明するまでもなく，サービスを受ける為に来た客（待行列理論の扱いでは一般にサービスを受けるものを客と呼ぶ）に対して，来た順番にサービスを行い，一つのサービスが完全に終了した後に次の客のサービスを行う方式である。この方式ではサービス時間の長いものも短いものも関係なく来た順番でサービスされる為に，ほんの短いサービスしか受けないのに，前に長い客がいるとずっと待たされると言った問題がある。計算機システムを考えた場合，短いジョブは早く処理すべきだと言う考え方がある。この様な条件を満たすべく考えられた方式が，このカンタム・タイム・サービス方式である。この方式では客に対するサービスをこま切れにして，各客に対してなるべく均一にサービスをする工夫をしている。一回に与えるサービス時間のことをカンタム・

タイムと呼んでいる。この様な方法でサービスを行えば，サービス時間の短い客は早くサービスを完了する事になる。

さて，カンタム・サービス方式を中心としたコンピュータ・システムの解析モデルは実に多い。ここでは，まづこれらのモデルを特徴付ける５つの観点，即ち

① スケジューリング・アルゴリズム

② カンタム・タイプ

③ オーバ・ヘッドの仮定

④ 到着分布

⑤ サービス・タイムの分布

の詳細を検討した後に今まで行われて来たこの分野に於ける仕事を分類していきたい。

A. スケジューリング・アルゴリズム

① Round-Robin（ＲＲ）

システムは１つだけQueue を持ち，新しく発生したジョブは，キューの最後にならぶ，システムは，キューの先頭のジョブをquntumだけサービスし，そのジョブが，まだ処理を続行する必要がある場合には新しく発生したジョブと同じ様にキューの最後にならぶ。この様子を次の図に示してある。

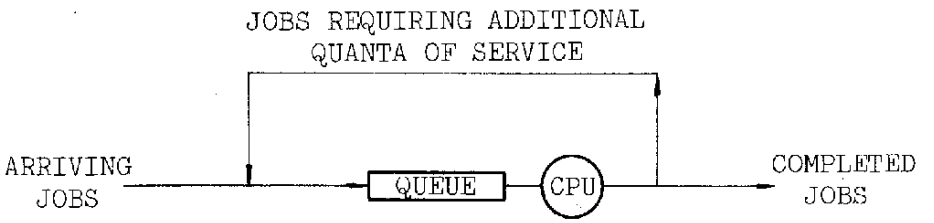


図４－27　Round-Robinスケジューリング方式

② Foreground Background（ＦＢ）

ＦＢのもとでは新しく到着したジョブ用のキューと，一度カンタム処理の終了したジョブがならぶＮ個のキューが用意されている。新しく到着し

たジョブがならぶキューをForeground キューと呼び，ジョブはキューの最後尾にならぶ。すなわちこのキューに関してはＦＣＦＳ（First-Come, First Sierviceベースで処理される。Foreground キュー のジョブはカンタム処理を１度終了した後さらに処理が必要であれば，Background キューと呼ばれる残りのＮ個のキューのうち第１番目のキュー（Ｎ個のキューは順番に番号が符られている。）にならぶ。以後，カンタム処理が終了する度に，第２，第３……第Ｎ番目のキューの順にならび，処理が終了するまで続けられる。

各キューにはプライオリティーが付けられていて，Foregroundキューが１番高く，第１ Background キューが２番目，以下順に下り，第Ｎ Background キュー に１番低いプライオリティーが与えられている。

１つのジョブに対するカンタム・サービスが終了すると，１番高いプライオリティーのキューにならんでいる先頭のジョブを選んでサービスを行う。即ち Foreground キューにジョブがあればこれを対象とするし，キューが空であれば第１ Background キューのジョブを，これも空であれば第２，と言う様に選ばれる。次の図はＦＢスケジューリング・アルゴリズムの概略を示したものである。

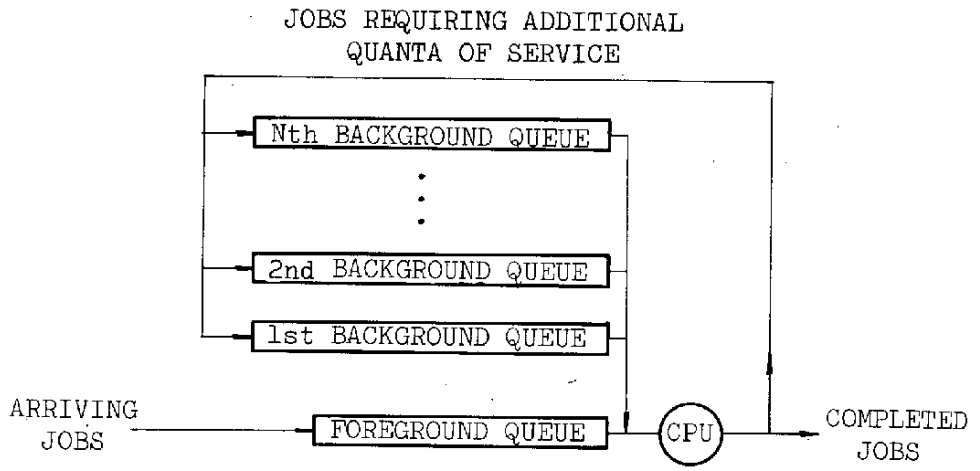


図４－28　ＦＢスケジューリング方式

ＦＢスケジューリング・アルゴリズムについては１つ問題が残されている。それは，第Ｎ Background キューで処理したジョブが，さらにカンタム・サービスを必要とする時にとる処置についてである。これには３つの方法が考えられる。

① 最低レベルのキューにあるジョブには続けてサービスをするが，もし途中で新しいジョブがシステムに到着するとカンタムが切れた所で新しいジョブのサービスを始める。さきほどのジョブは最低レベルのキューにそのままとどまるので順番がくれば再び同じ要領で続きのサービスを受けられる。

② 最低レベルのキューで処理が完全に終るまで，カンタム・サービス終了ごとに，キューの最後尾にならぶ事を繰り返す方法である。

③ 最低レベル・キューにあるジョブに対しては，無限大のカンタムを与え，そのジョブの処理が完全に終了するまでサービスを続ける方法である。

これらとはまったく別の考え方として，無限個の Background キューがあると考える法もあるが，これは，もちろん現実的でないが，取扱いの上から便利になる。

ＦＢに関しては，その Background キューの数Ｎに応じて，FB_N と表わし特に無限個のキューを持つものについてはFB_∞などと記述される。

ＲＲ，ＦＢスケジューリングの基本的な考え方は以上述べた通りであるが，さらに細い部分に手を加え，その相異によって次の５つに分類される。

① RR／D（Round-Robin with Delayed Entry）

これは新たに到着したジョブは一定時間経過したのちに Round－Robine キューに入る方式でKleinrock は "selfish round robin" と呼んでいる。

② FB／P（Foreground Background with Priority Entry）

新しく到着したジョブがフォアーグランド・キューだけでなく任意のバックグラウンド・キューにも直接入れる様にした点以外はRR／D

と同じ方式である。

③ FB／PP（Foreground Background with Priority Entry and Priority Service）

FB／Pに似ているが到着したジョブ自身もプライオリティーを持ち，同じキューにならんでいるジョブはこのプライオリティー順にサービスを受ける。同じプライオリティーを持つジョブ同しはＦＣＦＳベースで扱われる。

④ FB／PO（Foreground Background with Priority Entry and Oldest Job First Service）

FB／Pの特別なキューに対しては，そのキュー内のジョブのサービス順序はそのキューに到着した順でなくシステムに到着した順序でサービス順が決められる様な方式。

⑤ FB／NQ（Foreground Background with Non-Standard Queue Selection）

標準的なＦＢアルゴリズムではフォアーグランド・キューは１番高いプライオリティーを持ち，第１バックグランド・キューは第２番目，以下第ｎ番バックグランド・キューは第ｎ＋１番目のプライオリティーを持つが，FB／NQではこのプライオリティーの与え方を変更して扱う方式である。

B. カンタム・タイプ

カンタム・サービス方式はスケジューリング・アルゴリズムだけでなくカンタム・タイプにも幾つかの形態が扱われている。大きく分けるとカンタムの長さが確定値をとるか，確率変数で与えられるかの区別が出来る。しかし，確定値を取る場合にも毎回同じカンタムがすべてのジョブに適用され事を意味しているわけではない。

カンタム・タイプをさらに細かい基準で分類したものを次の表にまとめてある。

表4－22　カンタム・タイプの分類

記　号	説　明
DI	全てのジョブに対して一定長のカンタムが与えられる場合
RI	全てのジョブに対して同じ確率分布に従ったカンタムが与えられる場合
DP	ジョブを幾つかのクラスに分類し，各クラス内では一定サイズの同一カンタムが指定される場合（異ったクラスのカンタムはかならずしも一致しない。）
DN	カンタム長は，それまでに受けたサービスの回数などによって与える。FBアルゴリズムではバックグランド・キューのレベルに応じて与える。
RN	FBのキューのレベルに応じた確率分布でカンタム長が与えられる。
DPN	カンタム長は，ジョブ・クラス（DPと同じ考え方）と，それまでに受けたサービス回数，またはFBのキュー・レベルに応じた一定値で与えられる。
DD	カンタム長は，現在処理しているジョブ数とか，過去一定時間に到着したジョブ数などの関数として与えられる。
DIZ	タイプDIでカンタム・タイムを小さくして行き，ほとんどゼロになった状態で扱うもの
DPZ	同上（但しDIをDPで置き換える）
DPNZ	同上（但しDIをDPNで置き換える）

C．オーバ・ヘッドの仮定

ここでオーバ・ヘッドと呼んでいるのは，一つのジョブに対するカンタム・サービスが終了した　，ジョブがターミネイトした後に他のジョブにコントロールを渡すに要する時間である。カンタム・サービス方式ではほぼ4つに分類出来る。

表4－23　オーバ・ヘット

記　号	説　明
Z	オーバ・ヘッドをゼロとする。
C	オーバ・ヘッドは一定時間とする。
CPN	オーバ・ヘッドをジョブのプライオリティーやRRシステムでそれまでに受けたサービス回数または，FBシステムのキュー・レベルの関数として与える。
R	適当な確率分布に従った確率変数として与える。

D. 到着過程

一般の待行列問題の扱いとして到着パターンやサービス・パターンについて，Kendall が導入した記号は有名である。次の表には，quantum controlled sevice desciplinesに於て大事だと思われるもののみを拾って示してある。

表４－２４　到着分布

記号	名　　前	説　　明
B	Bernoulli Arrivals	各カンタムの終了後，新しいジョブが来るか否かのベルヌーイ試行を行う。各試行に於ける成否の確率は同じにする。
M	Poisson Arrivals	タイム・インターバルTの間にK個到着する確率が $\frac{(aT)^k}{K!}e^{-aT}$ で与えられる。（aは正の定数）
Mf	Finite Source Poisson Arrivals	CPUにあるジョブの数がjの時，次のジョブが到着するまでの時間が平均 1／a（N－j）（aは正の定数）の指数分布で与えられる。j＝Nになると次のジョブは到着しない。
GG	General Arrivals	到着時間間隔が一般の分布型で与えられるもの。

E. サービス・タイムの分布

ジョブが必要とするトータル・サービス・タイムを与える為の確率分布で到着分布と似た形で次の表の通り分類出来る。

表４－２５　サービス・タイム

記号	呼　び　名	説　　明
B	Bernoulli Sum Service Time	カンタム・サービスが終了する度にまだサービスが必要か否かを決る為にベルヌーイ試行を行い，「成功」が続く限りサービスを要求する。
M	Exponential Service Times	サービス・タイムを指数分布に従って与える。
G	General Service Times	サービス・タイムを適当な確率分布に従った確率変数で与える。
H	Hyperexponential Service Time	サービス・タイムをHyperexponential分布に従う確率変数として与える。

F. カンタム・サービス・モデルの分類

以上5つの項目のサブカテゴリーについて述べて来たカンタム・サービス・モデルは，これらのカテゴリーが指定されると一意にその構造が決定されてしまう。一度そのモデル構造が決れば，後は待行列理論に依って取扱う事が出来る。

次の図4－28，4－29はカンタム・サービス・モデルに於ける，これらのカテゴリーの関係を系統的に示したものである。

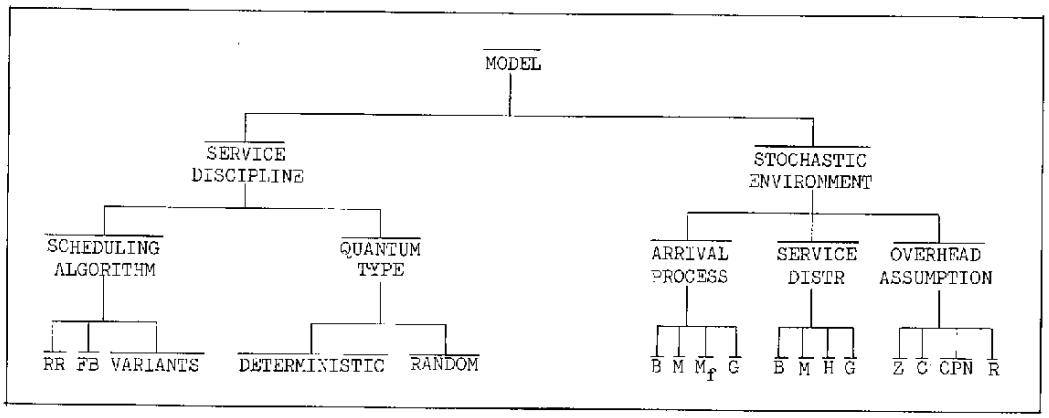


図4－29 Queueing Models of Quantum Controlled Service Disciplines

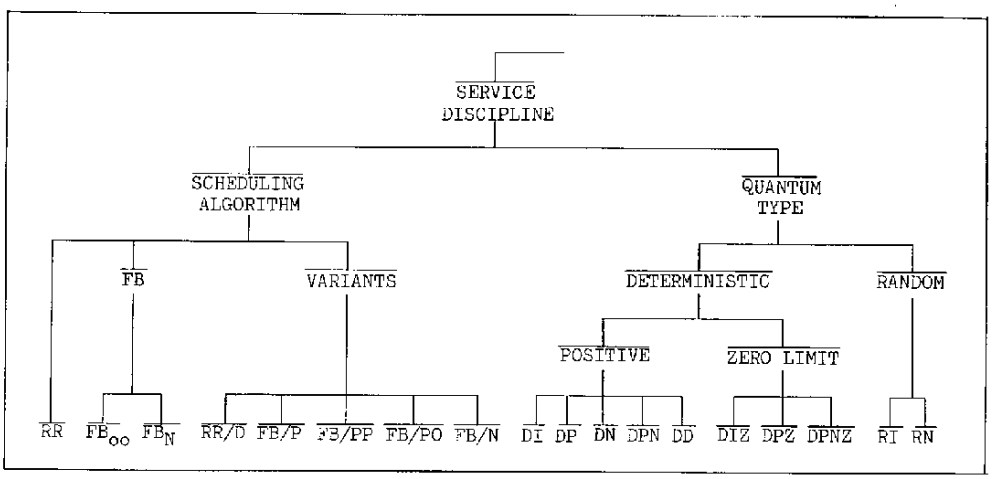


図4－30 Detailed Structure of the Service Discipline

カンタム・サービス・モデルの研究は数多く報告されています。最後にこれらを各カテゴリーに従って分類して，表にまとめ示しておく。

表4－26 Research and Survey Papers Dealing with the Analysis of Quantum Controlled Service Disciplines

AUTHOR (Ref. No.), DATE	SCHEDULING ALGORITHM	QUANTUM TYPE	ARRIVAL PROCESS/ SERVICE DISTR.	OVER-HEAD
Adiri & Avi-Itzhak (5),1969	RR	DI	M_f/M	C
Baskett (8),1970	RR	DIZ	M_f/H	Z
Chang (13),1966	RR	RI	M/B	Z,R
Coffman (17),1966	RR,FB_2,FB_N FB_∞/PP	DI,DN,DD DIZ	M/M, M_f/M	Z,C
Coffman (18),1967	--- SURVEY ---			
Coffman (19),1968	RR,FB_2	DI	M/M	C
Coffman (20),1968	RR,FB_2	DD	B/B	Z
Coffman & Kleinrock (22),1968	RR,FB_N FB_∞/P	DI,DIZ DPZ	B/B, M/M	Z
Coffman & Krishna-moorthi (23),1964	RR,RR/D	DI	M_f/B	C
Coffman & Muntz (24),1969	RR,FB_∞	DIZ	M/G	Z
Coffman, Muntz & Trotter (25),1970	RR	DIZ	M/M	Z
Estrin & Kleinrock (31),1967	--- SURVEY ---			
Fife (34),1966	FB_3/N	DN	M_f/H	C
Greenberger (44), 1966	RR	DI	M_f/M	C
Kleinrock (51),1964	RR	DI	B/B	Z
Kleinrock (53),1967	RR	DIZ,DPZ	M/M	Z
Kleinrock (54),1968	RR	DIZ	M_f/M	Z

AUTHOR (Ref. No.), DATE	SCHEDULING ALGORITHM	QUANTUM TYPE	ARRIVAL PROCESS/ SERVICE DISTR.	OVER-HEAD
Kleinrock (55),1969	ALL TYPES	DPN	G/G	CPN
Kleinrock (56),1970	RR/D	DIZ	M/M	Z
Kleinrock & Coffman (57),1967	ALL TYPES	DPN DPNZ	G/G	Z
Krishnamoorthi (59),1966	RR	DI	M_f/M	C
Krishnamoorthi & Wood (60),1966	RR,RR/D	DI	M_f/M	C
McKinney (61),1969	--- SURVEY ---			
Patel (64),1964	RR,FB_∞/PO	DI,DN	M_f/G, M/G	Z,C
Rasch (66),1970	RR	DI	M/M	Z,C
Sakata, Noguchi & Oizumi (68),1969	RR	DIZ	M/G	Z
Scherr (69),1965	RR	DIZ	M_f/M	Z
Schrage (70),1967	FB_{oo}	RN,DN DIZ	M/G, M/M	Z
Shemer (75),1967	RR,FB_∞/P	DI,DN	M/M	Z

4.3.2 Rotating Storage Service Disciplines

DrumとかDisk と言った回転型のストレッジ・デバイスの扱いは，言わば，待行列問題の典型的なものとして，多くの解析がなされている。この様なシステムを待行列問題として扱う場合，一般に，READ，WRITE 要求をカストマー，ディスクやドラムをサーバー，READ, WRITE処理に要する時間をサービス・タイムとするのが普通である。サービス・タイムのうちには，ドラムの場合，ヘッドが所定の位置に来るまでの回転待時間と，実際に情報の転送がなされる時間を含んでいるが，ディスクではさらにアームを移動する為のシーク・タイムも含まれている。その為，ドラム・システムの方が取り扱いがやさしいと言える。

A. ドラム・システム

ドラム・システムの場合ＦＣＦＳ（First Come First Service）方式はあまり効率的ではない。というのは回転待ち時間の間に実際は情報の転送が出来るＩ／Ｏ要求があるかもしれず，この時間を有効に利用する事が考えられていないからである。

DenningがＳＡＴＦ（Shortest Access Time First）と呼んだ方式が，かなり効率的であると報告されている。

この方式は，キューにあるＩ／Ｏ要求を適当な順番にならべなおして，隣り合った要求の回転待時間を最少にしようと言うものである。Weingarten, Denning, Coffman, Abate及びDubnerらがこの方式について調べた結果を報告している。始めの３人の著者は，各Ｉ／Ｏ要求の転送情報が一定長ブロックの場合について扱っている。Denning の場合には，待行列にある要求数の関数として，平均サービス・タイムを表わしている。WeingartenとCoffmanの場合には，Ｉ／Ｏ要求がポアソン到着するものと仮定し，平均到着率の関数としてシステム・パフォーマンスを求めている。Abate とDvbnerは可変長ブロックについて扱っているが問題がむずかしく，近似解しか得ていない。

B. ディスク・システム

ディスク・システムの場合にもサービス方式としてＳＡＴＦを適用する事が出来る。アームの移動を完了した後，同じシリンダーに対する要求に対してＳＡＴＦ方式が使われる。Weingartenはこの方式で十分のパフォーマンスが得られる事を解析し示している。しかし，他の多くの研究は各シリンダーに対してＦＣＦＳベースでサービスする方式について行われている。その理由はディスクのシリンダー数が十分に多い（例えば100や200有る場合）時には，よほど要求が頻繁に発生しない限り，同じシリンダーに対する要求が幾つもあってキューを作る様な事は，無いと思われるからである。

ディスク・システムの場合，ヘッドを析定のシリンダーに移動する為のスケジューリング問題がある。このスケジューリング方式についても，一番簡

単なものとしてＦＣＦＳが考えられるが，ヘッド移動についての考慮が欠けている意味で問題があると言える。

第２の方式としては，ＳＡＴＦ方式と同じ様な考え方で，やはりDenningが，ＳＳＴＦ（ Shortest Seek Time First ）を提案している。なるべくヘッドの動きを少なくしようとする方法ではあるが，むしろステップ・バイ・ステップで局所的に最少化する方式と言える。これに対してFrank はもっとグローバルな立場から最少化を計ろうとする事を考え，その時点でキューにある要求を全部調べて，トータルなヘッド移動が最少となる様に選ぶ方式を提案している。しかし，これらの方式にしても，問題が無いとは言えず，Ｉ／Ｏ要求が頻発した時などに，ヘッド位置が特定の場所に集中する傾向があり，ある要求に対してはなかなか順番が回らないと言った現象が見られる。この様な問題を避ける為にDenningはＳＣＡＮと呼ばれる方式を提案している。この方式ではヘッドをディスクの内側から外側へ，終ったら外側から内側へといつも二方向にだけ動く様な要求を選んでスケジューリングするものである。

彼は，この方式とＦＣＦＳ，ＳＳＴＦとを較べ，ＳＳＴＦが確かに一番効率的ではあるがＳＣＡＮ方式の方が好ましいと報告している。

ディスクのシーク・タイムを減らす別の方法として，データの配置を工夫する事も考えられる。例えば，頻繁に参照されるデータを中央に配置しておけば，ヘッドが，この位置にある確率が高くなるので，ヘッドは悪くとも半分の距離を移動すれば済む事になる。Frank, Abate, Duben, Weinbergらは，この様なデータ・アロケーションに依る効率の変化についても調べている。

4.3.3　プライオリティー・モデル

今まで扱って来たモデルは，キューに対するプライオリティーが与えられているもの（例えばＦＢ）はあってもジョブが個有のプライオリティーを持つ様なものはなかった。一般のリアル・タイム・システムを例に取っても，ジョブ自身にプライオリティーを持たせるのが有効である事は理解出来る。この様に

プライオリティーを持つジョブを扱ったモデルをプライオリティー・モデル（Conventional Priority Disciplines）と言う。プライオリティー・モデルは次の３つの項目で特徴付ける事が出来る。

(1) ジョブはシステムに到着した時からプライオリティーが与えられる。

(2) 一番プライオリティーの高いものからサービスを受ける。

(3) ジョブのプライオリティーは途中で変らない。

Conventional Priority Disciplinesは前に述べた性格の２番目の解釈の仕方に依っていくつかの方式に分ける事が出来る。例えば，サーバーは一つのジョブを完全にサービスし終らないと次のジョブのサービスを行わない場合これを非割込処理方式と呼んでいる。この方式では途中で，処理中のジョブより高いプライオリティーを持ったジョブが入って来ても，完全に処理が終るまでこのジョブのサービスを開始しない事になる。

一方，途中で高いプライオリティーのジョブが来たらいつでも，現在のサービスを中断して，そのジョブのサービスを行うようにする。この方式を割込処理方式と呼ぶ。この方式については，途中で処理をやめたジョブのその後の扱い方に依って，さらに分類する事が出来る。

中断された地点からサービスを続けるものをpreemptive-resum descipline と呼んでいる。また，中断されるまでに行われたサービスは無視して，サービス開始時点からやりなおす方式をpreemptive-repeat-identical と呼んでいるが，この方式はあまりコンピュータ・システム向きとは言えない。コンピュータ・システムの処理自体，中断したジョブの処理は，その時点から再開する方がむしろ簡単である。この方式では，さらに再開後のサービス・タイムを中断前に与えられたものをそのまま使うのでなく，新たに計算しなおす場合が考えられる。これをpreemptive-repeat identical と区別してpreemptive-repeat-different と呼んでいる。

しかし，この方式もコンピュータ・システムの解析に用いられた例はあまり見られない様である。

これらの方式がコンピュータ・システムの解析に用いられた例としては，初

期の頃，non-preemptive priority discipline が Chang と Wong に扱われた例がある。さらに新しい所では，やはり Chang が，non-preemptive やpreemptive-resume disciplineを適用した例が報告されている。

以上のモデルでは同じプライオリティーを持つジョブの間では到着順，即ちＦＣＦＳベースでサービスがなされるが，これらのジョブに対してカンタムに依るサービス方式を行う事も出来る。例えば Chang は，同じプライオリティーを持つジョブのグループごとに，ＲＲ，ＲＩ，Ｍ／Ｂ，Ｚシステムを適用している。また，AdiriもＲＲ，ＤＩ，Ｍ／Ｍ，Ｃシステムを適用している。これらはいずれも，異ったプライオリティーのジョブ間ではpreemptive-resumeタイプとして扱っている。Adiri は，その後，同じグループ内でのサービス方式について３つの新しい方式を報告している。その第１は，一たびジョブにカンタムが割当てられると，それが終了するまでは割込みを禁止し，サービスが終った時点で，始めて割込みを起す方式である。

第２は，割込みはいつでも可能とし，割込まれて中断されたジョブでは，そのカンタム・タイムの始めからやりなおす方式で，これは丁度，preemptive-repeate-identical と preemptive-resume の中間的な方式と言える。

第３の方式は，第２の方式と，普通の preemptive-resume descipline の中間的な形で，割込みはどんな場合にも発生するが，次のカンタムを与える為の処理中に発生した場合には割込まれたジョブのカンタムは廃棄され，再開する時には，またそのカンタムの始めから処理を続ける。カンタム処理途中で起った割込みについては，割込まれた地点から再開される。

Schrage の示したモデルでは Adiri のものとほとんど同じであるが，各プライオリティー・グループ内でのサービス方式がＦＣＦＳとなっている所だけ異っている。彼のモデルではジョブの３つのグループに分けて考え，各々について①non-preemptive，②preemptive-resume，③preemptive-repeat interval の扱いをしている。

4.3.4 Capacity Problems

これまで取扱って来た解析は，全て待行列について，リスポンス・タイムや

待時間を決定する問題であった。この様に時間に関連した問題以外にコンピュータ・システムの解析を行うにはスペースの問題を取扱はなくてはならない。

例えば，ジョブが必要とするストーレッジの問題，キューの長さに制限があるシステムでキューがオーバフローする時間間隔を求める問題などがその例である。

この様な問題を総称して＂Capacity Problem＂と言っている。BoudreamとKac，Harrison，Weingerten，ChemgとWovg，Cam，Bowdonらの報告が見られる。

これら論文の中では，グループとかバッチ到着を取扱っているものがある。この場合，その時点で到着したカストマーの人数を表わす確率変数が新たに導入されている。しかし，到着間隔は，従来通りに取扱われているので，通常のキューイング・システムは，バッチ到着システムの特別なケースとして考える事が出来る。バッチ到着モデルを扱った初期の研究例としてBoudreauとKacのモデルを挙げる事が出来る。

このモデルは，n秒ごとにトランザクションのグループを発生するインプット・ジェネレータを想定し，各グループのトランザクション数は整数値を取る確率変数で与えられている。各トランザクションの処理に要する時間は発生間隔の2倍，つまり2n秒かかると仮定している。この様な条件のもとでシステム定常状態に於けるトランザクション数の分布をMarkovチェインを使って計算している。また，使用可能なストーレッジに制限を加えて，キューがオーバフローする間隔の平均時間も求めている。

Weingertenの場合，同じ様にバッチ到着を仮定した。コンピュータのメッセージ・スイッチ・モデルを扱っている。このモデルは，1つのプロセッサーに対して，n本のラインからメッセージが送られてくると，プロセッサーは文字単位に分解して送り返すものである。1文字の転送に要する時間は一定値で扱われているのでメッセージの返送時間は文字数に比例した値となる。

プロセッサーに接続されている各ラインからはメッセージがポアソン分布に従って到着し，そのメッセージを送り返すに要する時間は指数分布に従った確

率変数で与えられる。この様な仮定のもとでWeingartenは各文字をカストマーとして，メッセージはカストマーのバッチで，そのサイズは指数分布で与えられ，各カストマーのサービス時間を一定値とする待行列モデルを考えた。彼のモデルの利点は，任意時刻に於て，システムに存在する文字数が，到着したメッセージの待時間に，単位時間当りの転送可能な文字数を乗じた値に等しい事である。到着メッセージの待時間はポアソン到着，指数サービスの様な簡単なケースでは計算する事が出来る。さらにWeingartenは，使用可能なストーレッジ（ storage ）のサイズを有限値としてキューがオーバフローする平均時間間隔を求めている。しかし，Boudreau やKacの場合よりはむずかしい問題なので，いくつかの単純化をしており，近似的な解しか得ていない。

Harrison も Weingarten と同じ様なコンピュータのメッセージ・スイッチング・モデルを考察しているが，その相異は，Harrisonの場合，容量(Capacity ）を文字に対してでなくメッセージに対して考えている点である。しかし，この点を除くとWeingarten のモデルと非常に良く似ている。Harrisonはこのモデルを使って，各ラインごとに固定サイズのバッファーを持つ場合と，全てのラインに共通なバッファーを分割して使う場合のパフォーマンスを比較している。

Boudreau，Kac，Weingarten らのバッチ到着型モデルでは各カストマーのサービス・タイムは一定値で与えられているが，もちろんこれを確率変数で与える事も出来る。Delbrouck は，この様なモデルを考案し，サービス・タイムを決める為には複雑な確率関数を使っている。このモデルはコンピュータ・システムに於けるインプット・ラインのポーリング問題を解析するのに用いられた。

Chang と Wong はこのキュー容量問題で少し異ったアプローチをしている。Chan は各ジョブをストアーするのに必要なスペースの総数を母関数F（ Z ）で決る確率変数を使って指定している。そこで，もしキューにあるジョブの数を決める母関数をU（ Z ）とすると，キューにいる全部のジョブが必要とするストーレッジは母関数U（ F（ Z ））を持つことになる。このモデルの良い点

はU（Z）が，一般に，わかり安い方法で求められる場合が多い事である。従ってF（Z）とさえ与えられればジョブが必要とするストーレッジの総量がすぐに求まる事になる。

Bowdonは conventional priority service disciplinesのキューイング・システムでキューサイズに制限がある場合を調べている。彼のモデルでは，R個のプライオリティーを持ち，K種類のサービスを1つのサービス・ファシリティーによって，非割込み方式で行うものである。各プライオリティーのジョブがシステムにポアソン到着し，サービス時間はどのジョブも同じ平均値を持つ指数分布で与えられている。

Bowdonは，このどちらかと言えば一般的なモデルに制限条件を加えキューにはMジョブしか待つ事が出来ず，もし，丁度Mジョブキューに待っている状態で，新しいジョブが到着すると，自分よりプライオリティーの低いジョブがキューに有る時に限って，そのジョブと入れ換わってキューに入る事が出来る様にした。この場合入れ換えられたジョブは呼損として扱われる。同様に，もし新しく到着したジョブよりプライオリティーが低いジョブがキューに無ければ到着したジョブ自身が呼損として扱われる。このシステムについて，Bowdonは，キューに入っているジョブの平均数をプライオリティーごとに求めた。また，各プライオリティーごとのジョブの平均待時間，ジョブが待っている間に自分より高いプライオリティーのジョブが来て，入換えられる事がない様な確率も求めている。

4.3.4　ネットワーク・モデル

これまで取扱って来たモデルは全て，ＣＰＵ，ディスク，ドラムなどが単独で作るシステムばかりであった。

しかし，実際のコンピュータ・システムは幾つものリソースがお互に関連しあって出来るシステムである。この様な意味で，これまでのモデルはコンピュータ・システムの解析を十分に行い得たとは言いがたい。

ネットワーク・モデルと言うのはこれらいくつものリソースを関連付けて取扱おうとするものである。しかし同じネットワーク・モデルと言ってもその中

にはやはり，色々なモデルが扱われており，対象とするリソース，構成，扱い方などさまざまである。ここでそのいくつかを紹介してみる事にする。

(1) File と Rosenberg のモデル

彼らのモデルは，プログラム・ローディング時間が，実行時間に及ぼす影響について調べる為のものであるが，モデル自身については色々と批判もある。むしろ，コンピュータ・システム・アナリシスの分野で，キューイング・ネットワーク・モデルの可能性を示した事の方が評価されている様である。

(2) Smith のモデル

彼は，さらに興味深い可能性を示してくれた。2つのモデルについて論じ，1つはかなり複雑でもあるが実際的なモデルと，これよりはいくぶん単純化し，取扱いを楽にしたモデルを示している。この第2のモデルはＣＰＵ，disk，有限個のインタラクティブ・ターミナルからなり，プログラムはディスクからロードされ，指定された時間だけＣＰＵを使うと言ったものである。プログラムが一定時間のＣＰＵを使い終ると，確率p,q,r（但し，p＋q＋r＝1）に従って

① ターミナルに対してインプットを要求する。

② ディスクからオーバレイ・セグメントをロードする。

③ プログラムをターミネクトし，次のプログラムをディスクからロードする。

のいづれかの処置がほどこされる。彼のモデルでは，オーバレイの為にセグメントをロードするのと，新しいプログラムをロードするのに同じＩ／Ｏチャネルを使用している為，このサービス順序を指定する必要がある。この為，彼は2つの方法について調べている。

第1は，新しいプログラムのローディング中に割込みせず，ただセグメントのロードを優先的に扱う方法，第2は，プログラムのロード中にセグメント・ロードの要求がくると割込み処理をし，セグメント・ロードが完了した後に，プログラム・ロードを始めからやりなおす方法である。

また，彼のモデルで興味深い点は，プログラム・ローディング・タイムの

分布として指数分布だけでなく，２次のアーラン分布も用いている点が挙げられる。これは，それまでの扱いがほとんど指数分布だけであったのに対して新しい分布の取扱い可能性を示している。

(3) Kleinrock

彼は始め２つのプロセッサーを持つシステムを考えた。ジョブはシステムに到着すると第１のプロセッサーでサービスを受け，インター・プロセッサー・バッファーに入った後，第２プロセッサーでサービスを受け，処理を終了する。

インター・プロセッサー・バッファーには，その個数に制限を加え最大N個までのジョブが入り得るとしてある。

インター・プロセッサー・バッファーが一パイになると第１プロセッサーは処理を中断する。従ってシステムパフォーマンスは低下する事になる。バッファーが一パイに成る確率はサービス時間（指数分布と仮定している。）と，バッファー個数Nに依存するが，Kleinrock はこれらのパラメータを系統的に変えて，システム・パフォーマンスに及ぼす影響を調べている。

さらに彼は，プロセッサーの個数をM個までに拡張し，ジョブは第１プロセッサーからあいだにあるインター・プロセッサー・バッファーを経由し，順に第Mプロセッサーまでサービスを受け，その後システムの場から立ち去ると言ったモデルを考えている。しかし，かなり近似的な解しか得ていない様である。

(4) Gaver

Gaverはさらにむずかしいネットワーク・モデルの解析を行い報告している。彼のモデルはＣＰＵと複数個のＩ／Ｏプロセッサー，同じく複数個のプログラムから成っている。プログラムはＣＰＵ処理を終了すると，続けてＩ／Ｏ要求を出し，これを繰り返す。Ｉ／Ｏ要求に対する処理時間は指数分布で与えられ全て同じ平均を持つとし，現在処理しているプログラムの個数が，使用可能なＩ／Ｏプロセッサーより多くならない限り，Ｉ／Ｏ要求に対する待行列は出来ないものとして扱っている。このモデルについて，彼は，プロ

グラム数とＣＰＵ利用率の関係を求めている。この仕事がユニークなのは，式の扱でＣＰＵサービス・タイムの分布型をexplicitに指定せず，最終的に，分布型が，ＣＰＵコティリゼーションを決める式の中でだけあらわされている事である。その為に異った分布型に対する評価が簡単に出来るメリットがあげられる。

このGaverのモデルはfinit sorce time sharing modelと非常に似かよっており，事実，Ｉ／Ｏプロセッサーの数とプログラムの数を同じにすれば，指数分布の"think time"，サービス方式がＦＣＦＳ及び一般型サービスタイムのfinit-source-time-sharing modelに成ってしまう。

参 考 文 献

1) Deniston, W. W. "SIPE: A TSS/360 Software Measurement Technique" Proceedings ACM National Conference (1969), 229-245.

2) Sherman, S., Browne, J. C. and Baskett, F., "Trace Driven Modeling and Analysis of CPU Scheduling in a Multi-programming System" Proc. of ACM Workshop on Performance Evaluation, Cambridge, Mass. pp. 173-199, (April, 1971).

3) Grenander, U. and Tsao, R. F. "Quantitative Methods for Evaluating Computer System Performance: A Review and Proposals" IBM Report RC 3478, Yorktown Heights, N.Y., July 27, 1971.

4) Oppenheimer, G., and Weizer, N. Resource Management for a Medium Scale Time-Sharing Operating System. Comm. ACM 11, 5 (May 1968), 313-323.

5) Denning, P. J. The Working Set Model for Program Behavior. Comm. ACM 11, 5 (May 1968), 323-333.

6) Fine, G. H., et al. Dynamic Program Behavior Under Paging. Proc. ACM 21st Nat. Conf., 1966. Thompson Book Co., Washington, D.C., 223-228.

7) Cantrell, H. N., and Ellison, A. L. Multiprogramming System Performance Measurement and Analysis. Proc. AFIPS SJCC, vol. 32, 1968. Thompson Book Co., Washington, D.C., 213-221.

8) Denning, P. J. Thrashing: Its Causes and Prevention. Proc. AFIPS SJCC, vol. 33, 1968. Thompson Book Co., Washington, D.C., 915-922.

9) Karush, A. D. "Two Approaches for Measuring the Performance of Time-Sharing Systems," Proceedings 2nd Symposium on Operating System Principles, Princeton, N.J., Oct. 1969, pp. 159-166.

10) Winograd, J., Morganstein, S. J., Herman, R. RCA Corporation, Computer Systems Division, Cinnaminson, New Jersey "Simulation Studies of A Virtual Memory, Time-Shared, Demand Paging Operating System"

11) Murty Parupudi, Addressograph Multigraph Corp. Joseph Winograd, Control Data Corp. "Interactive Task Behavior in a Time-Sharing Environment" Proceedings ACM National Conference (1972), 680-692

12) Scherr, A. L., "An Analysis of Time-Shared Computer Systems", Research Monograph No. 36, M.I.T. Press, Cambridge, Mass., 1967.

13) DeMeis, W. M. and Weizer, N., "Measurement and Analysis of a Demand Paging Time-Sharing System", Proc. 24th Nat. Conf. ACM, ACM Pub. p-69, 1969, 201-217.

14) Wulf, W. A. "Performance Monitors for Multi-Programming Systems" Proc. of the 2nd Symposium on Operating Systems Principles, Princeton, N.J., October 1970.

15) Lucas, H. C. Jr. "Performance Evaluation and Monitoring" Computing Surveys 3, 79-92 (1971).

16) Pinkerton, T. B. "Performance Monitoring in a Time-Sharing System" Comm. ACM 12, 608-610 (1969).

17) Arden, B. and Boettner, D. "Measurement and Performance of a Multi-Programming System" Proceedings of the 2nd Symposium on Operating Systems Principles, Princeton, N.J., Oct. 1970.

18) Morganstein, S. J., Winograd, J., and Herman, R. SIM/61: A Simulation Measurement Tool for a Time-Shared, Demand Paging Operating System. Proc. ACM Sigops Workshop on System Performance Evaluation (April 1971), 142-172.

19) Schwetman, H.D., Boole and Babbage, Inc., Brown, J.C., University of Texas, Austin. "An Experimental Study of Computer System Performance" Proceedings ACM National Conference (1972), 693-703.

20) Charles Gilbert Moore III. "Network Models for Large-Scale Time-Sharing Systems" 1971.

21) Herbert Dewitt Schwetman, Jr., B.S., Sc.M. "A Study of Resource Utilization and Performance Evaluation of Large-Scale Computer Systems" August, 1971.

22) Jeffrey P. Buzen "Queueing Network Models of Multiprogramming" August, 1971 ESD-TR-71-345.

5．コンピュータ・ネットワークとその評価

5. コンピュータ・ネットワークとその評価

システムの作成は，設計，インプリメンテーション，実測，評価の一連のプロセスを経て完成される。

しかし，システムの規模が大きくなるにつれ，必らずしも一通りの経過で最適のシステムを完成させることはむづかしい。評価の結果，当初目標としたシステム基準に達していない場合，あるいはより改良の予地があると推定される場合は，再三再四，設計，インプリメンテーション，実測，分布型プロセスが繰返されることがあり得る。

コンピュータ・ネットワークは，多少個々の規模の差はあるにしても，現在のシステムの中では少なくとも大規模システムの範中に含まれるものが多く，特に評価の必要性が高く，且つシステムの改良のためのサイクルが不可欠なものの一つである。

コンピュータ・ネットワーク・システムは米国に於てすらまだ実用経験が浅く，一般のコンピュータ・システムに比べまだ評価技術も確立してはいないが，米国のＡＲＰＡネットワークを中心に，現在までのネットワーク評価の技術と実例をまとめてみたい。

まず，第１節では，ネットワーク・システムの事例を紹介し，それぞれのシステムの比較を表に示す。

第２節では，コンピュータ・ネットワーク設計技術と問題点を中心に，ネットワーク設計で利用される道具も紹介する。

第３節では，ネットワークの測定道具と測定結果を紹介する。

最後に，第４節では，ネットワーク制御，プログラムの改良と，その改良されたシステムのパフォーマンスの測定結果を紹介する。

5.1 コンピュータ・ネットワークの概要

コンピュータ・ネットワークは，各地に散在する独立したコンピュータを高速回線で接続し，ハードウエア，ソフトウエア（プログラム），データ等のリソースを共同利用することを目的として生まれたものである。

コンピュータ・ネットの主な利点としては，

(1) ロードの分散

あるコンピュータ・センタが飽和状態のとき，センタ・ジョブを別のセンタへ回すことによって，ネットワークに参加している各センタの負荷を均等化することが可能になる。

(2) データの分散

大量なデータを擁するデータ・バンク等において，絶えず更新を要求されるデータを各コンピュータ・センタごとに維持することは不経済である。ユーザは，自分のプログラムを他のシステムへ送り込んで，そのシステムにあるデータを利用したり，あるいは，データそのものを送ってもらって利用する。

(3) プログラムの分散

ネットワーク中のどの地点からも，どのコンピュータへもアクセスできるようになると，各センターは，それぞれ得意の分野のプログラムを開発すればよく，プログラム開発の重複投資がさけられる。

コンピュータ・ネットワークの形態としては，概略，図５－１のようなものが考えられる。すなわち，各ホスト・コンピュータ間を結ぶ高レベル・ネットワークと，ホストと端局間を結ぶ低レベル・ネットワークの２種類のサブ・システムで形成されている。

コンピュータ・ネットワークに参加しているコンピュータが，すべて同機種（オペレーティング・システムが同じという意味を含む）で構成される方式と，異機種のコンピュータによって構成される方式がある。

同機種コンピュータ・ネットワークは，異機種ネットワークとくらべて，費

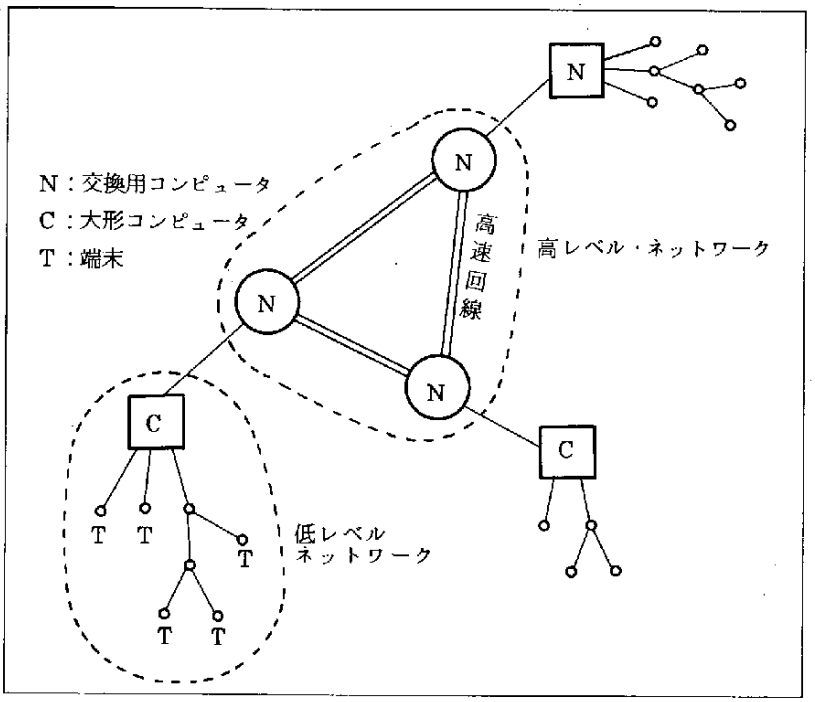


図5－1　コンピュータ・ネットワークの形態

用も安価ですみ，しかも，短期間で開発をすることが可能である。そのうえ，コンピュータ・ネットワークの特徴である，ロードの分散とか，プログラムの交換を比較的容易におこなうことができる。

しかし，一方，特殊なハードウエアおよびソフトウエアでのみ処理できる作業の実行は，異機種コンピュータ・ネットワークの設立によって，はじめて可能になる。

以下に，いくつかの代表的なコンピュータ・ネットワーク・システムを1つ1つ簡単に紹介する。

これらのシステムの比較を表5－1に要約してある。

＊1. 米国の電話会社やウエスタン・ユニオンが提供している専用線。

＊2. Direct distance dialing（自動即時通話方式）：電話使用者が交換手によらないで他の加入区域の加入者を呼び出せる電話交換サービス。英国とその他数カ国では，"Subscriber Trunk Dialing（STD）"と呼んでいる。

表5－1　コンピュータ・ネットワークの比較

比較項目＼ネットワーク名	ARPA	CYBERNET	DCS	MERIT	NETWORK／440	NPL	OCTOPUS	TSS	TUCC
稼動開始日	1969	—	—	1969	1968	1970	1971	—	1965
ネットワークの形態	分布型	分布型	分布型	分布型	集中型	分布型	集中型 ① 分布型 ②	分布型	集中型
コンピュータの構成	異機種	異機種	異機種	異機種	異機種	異機種	異機種	同機種	同機種
ノード数	34	36	9	3	12	27	10	9	4
ノードの場所	アメリカ	アメリカ	カルフォルニア大学（アービン）	ミシガン州	ヨークタウン	NPL	LBL	アメリカ	ノース・カロライナ州
計算機の規模	複合	大	ミニ	大	大	大	大	360／67	IBM 360
通信交換用マシン	Honeywell DDP 516,316	wideband swith	リング・インタフェース	PDP−11／20	IBM 360／91	DDP−516	PDP−6	IBM 2701	IBM 2701
回線形態	交換回線	交換回線	交換回線	交換回線	交換回線	直結回線 交換回線	直結回線① 交換回線②	交換回線	直結回線
交換方式	蓄積交換	回線交換	リング交換	蓄積交換	蓄積交換	蓄積交換	蓄積交換	蓄積交換	—
伝送媒体	賃貸回線	賃貸回線	同軸ケーブル	Telpak*1	貸借半2重回線	同軸ケーブル	同軸ケーブル	DDD*2	Telpak*1
データ伝送率（bps）	50,000	40,800	200−500キロ	2,000	40,800	1,000キロ	12メガ	2,000 40,800	40,800
伝送モード	アナログ	アナログ	デジタル	アナログ	アナログ	デジタル	デジタル	アナログ	アナログ
メッセージ・フォーマット	可変長	固定長	固定長	可変長	—	固定長	可変長	可変長	可変長
メッセージの大きさ（最大）	8,095 ビット	1,024字	900ビット	240字	—	1,024 ビット	①3,780キロビット ②1,208ビット	8,192 ビット	1,000 バイト
備考							①：ファイル伝送 ②：テレタイプ		

A．ARPAコンピュータ・ネットワーク

1968年の末に，米国国防省のAdvanced Research Project Agency（ＡＲＰＡ）は，各地に散在している大学，研究機関にある多数の異機種コンピュータを共用搬送回線網*（common-carrier circiuts）で接続する新しいタイプのコンピュータ・ネットワークのインプリメンテーションに着手した。このような接続の当初の目的は，リソース・シェアリング（resource sharing ）である。すなわち，ある研究所にいる人とか，プログラムとかが，ネットワーク中の別のコンピュータにあるデータをアクセスしたり，プログラムをランさせ，インタラクティブに利用することである。ネットワークは，50Kbpsの高速回線とメッセージ交換技術によって実現された。

ＡＲＰＡネットワークはすでに3年間以上も稼動しており，国家的な施設となった。ネットワークは，米国全域に渡って30サイト（site：設置場所）以上にもなり，なお，拡大中である。各種のメーカーの40以上もの独立したコンピュータ・システムが相互接続され，独立したコンピュータ・システムをもたないサイトからでも，端末からネットワークをアクセスできるようになっている。

1969年の小さい4ノード・ネットから1972年末の34ノード・ネットまでのネットワークの発展過程を図5－2に示す。図5－3は図5－2(e)を詳細に表わしたものである。

図5－3からわかるように，ＡＲＰＡネットワーク中の各々のサイトは，4台までの独立したコンピュータ・システム（ホストと呼ぶ）とインタフェース・メッセージ・プロセッサ（Interface Message Processor，or ＩＭＰ）と呼ばれる1つの通信用プロセッサから成っている。各サイトにあるすべてのホストは，ＩＭＰに直結している。いくつかのＩＭＰは，ネットワークにホストを介さず直接端末を接続できる機能をもっている。これらは，ターミナル・インタフェース・メッセージ・プロセッサ（Terminal

---

＊高速電話・データ用回線

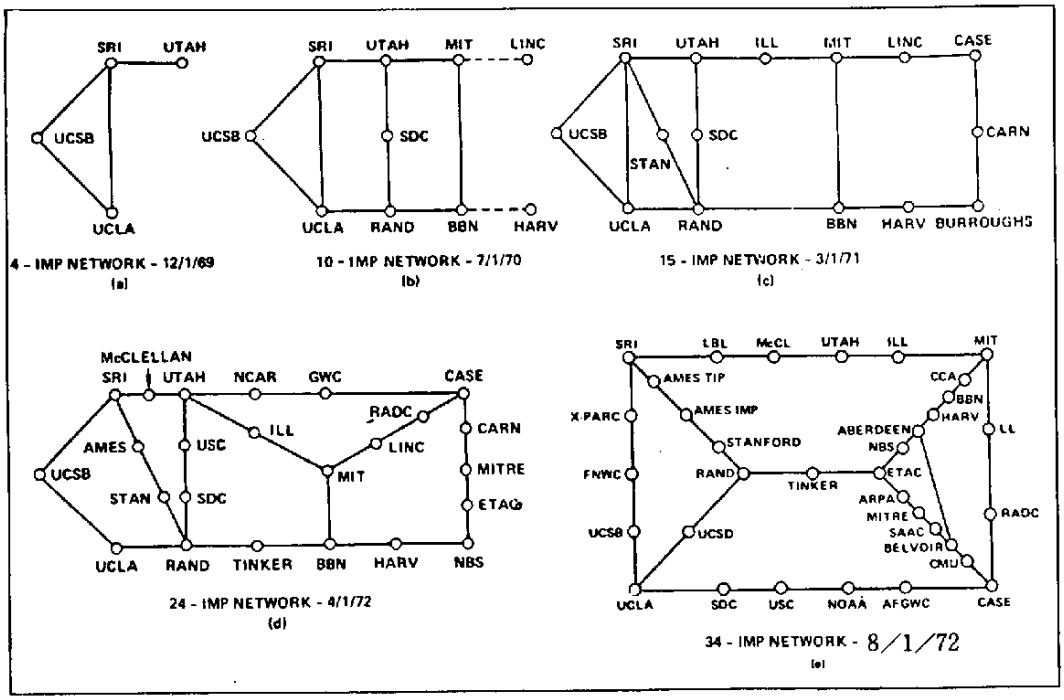


図5－2　Evolution of the ARPA Network

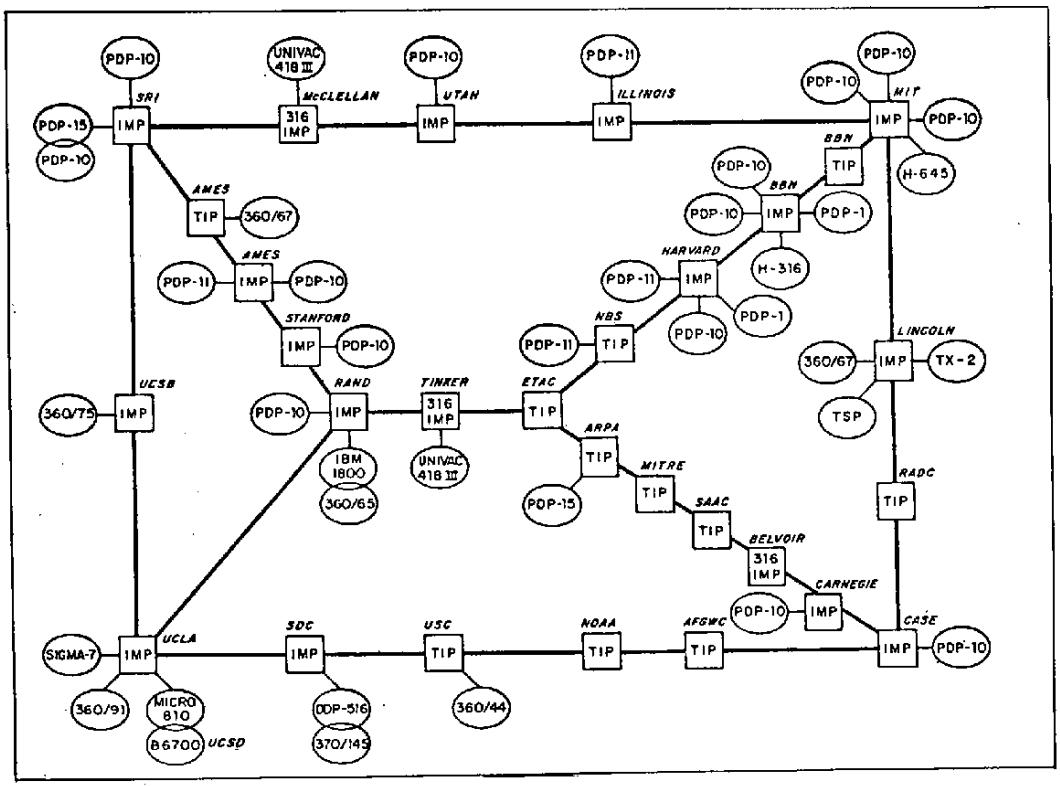


図5－3　ARPA NETWORK, LOGICAL MAP, AUGUST 1972

Interface Message Process, or TIP）と呼ばれる。IMPは高速電話回線に接続され，ホスト同志で通信できるようにサブネットを構成している。各IMPは，9.6～230.4キロビット／秒の伝送速度をもつ電話回線で接続可能で，5つまでのIMPと結合できるようになっている。現在のネットワークの伝送速度は，50キロビットである。

IMPの主な役割りは，ホスト・コンピュータとネットワークのインタフェースで，メッセージの蓄積交換を行なっている。ホスト間の通信は，一連のメッセージの送受によって行われている。メッセージは，ホストからIMPへセグメント単位で送られ，セグメントはIMPで規格化されたパケットに構成され，パケット毎にIMPからネットワークに送り出される。パケットは目的地IMPで再び集められ，メッセージに再組立てして，目的地ホストへ順次送られる（図5－4参照）。したがって，ホストはメッセージの送受信のみを取扱い，ネットワーク上でのメッセージの分割などに対してはまったく無関与でいることができる。

加えて，IMPは，メッセージの伝送時間を最小にするためネットワークを通るメッセージの経路選択を行ない，回線利用率を高めている。

各々のホスト・コンピュータには，NCP（Network Control Program）と呼ばれるプログラムが各OSに組込まれている。

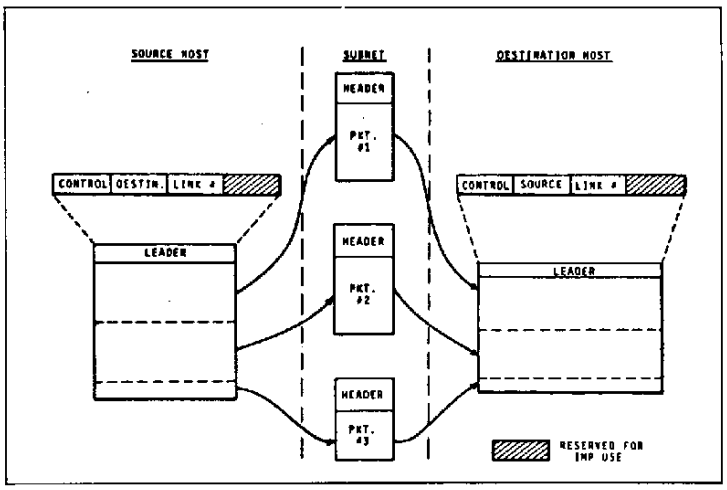


図5－4　Messages and Packets

ＮＣＰの機能は，ネットワークとホスト内のプロセス間のインターフェースである。ＮＣＰは，あるホストにあるプロセスと別のホストにあるプロセス間の結合を開設したり終結したり，ホストが結合されている方式とか，どのＩＭＰと通信しているとか，いうネットワークのインプリメンテーションの詳細に関与することなしに，情報の交換ができるようになっている。

B．CYBERNETコンピュータ・ネットワーク

コントロール・データ社のデータ・サービス・ネットワークCYBERNETは，最初の商業ベースのコンピュータ・ユティリティ・ネットワークである。これは，広範囲のネットワークではないが，すでに，以下の２つのサブネットワークが稼動しており，その利点が認められている。米国の西部／中西部サブネットワークと東海岸サブネットワークは，お互に結合し，さらに，南部で稼動している広く放射状のシステムに結合している（図５－５）。

東海岸ネットワークは，３つのＣＤＣ6600 システムと，２つのＣＤＣ3300 システムを高速伝送回線と交換で，お互の，あるいは自分自身エリアの放射状システム・サービスをお互にサポートすることによって，相互に連結している（図５－６）。西部／中西部ネットワークは，同じ機能をもった同じコンピュータ・システムからなっている（図５－７）。

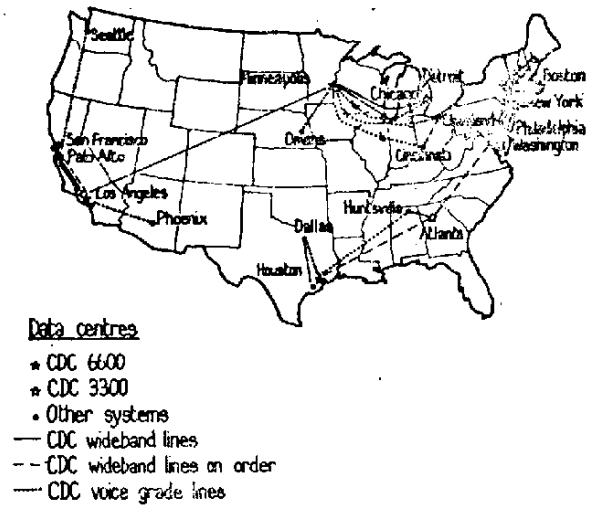


図５－５　CDC CYBERNET network

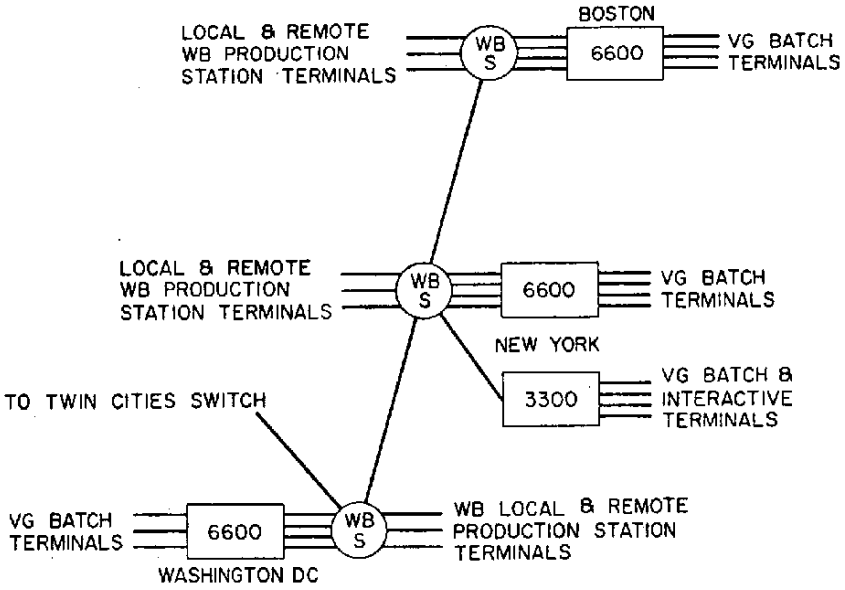


図5－6　Eastern Seaboard Segment of CDC's Cybernet

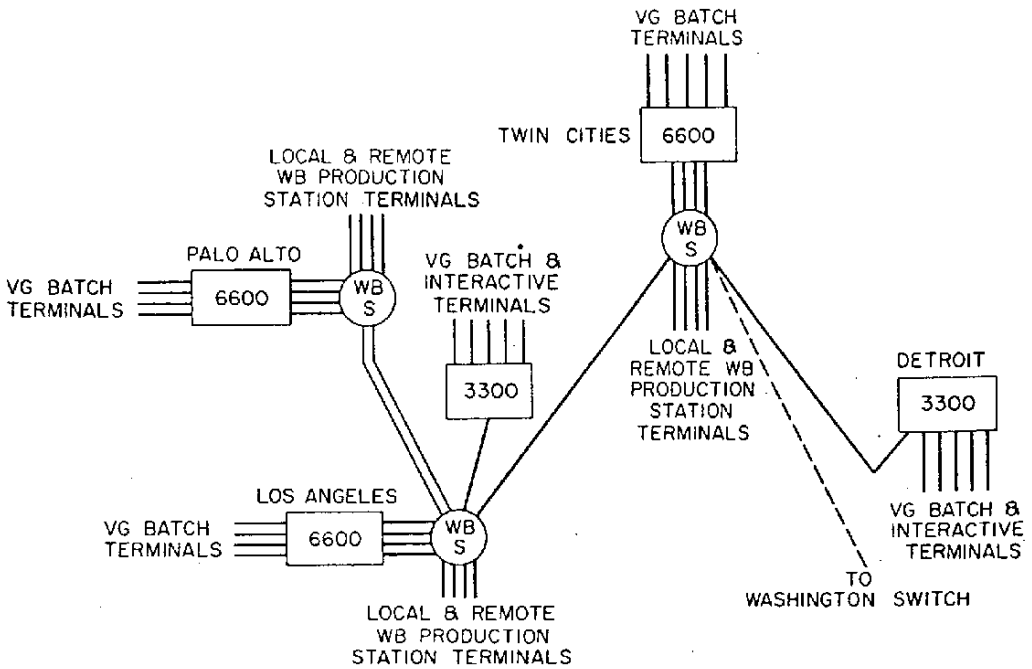


図5－7　West／Midwest Network Segment of CDC's Cybernet

CYBERNETは，５つの主要構成要素，すなわち，ユーザ，端末，通信回線，ノードそしてセントロイド・システム（Centroid：コンピューティング機能をつかさどるホスト）から成っている。

これらの要素の基本的な関係を，図５－８に示す。ユーザは端末を介して，別の端末ユーザと，ノードと直接又は別のノード経由でノードと，ノードを経由して別の端末ユーザと，セントロイドと直接，あるいは，ノード経由してセントロイドと通信する。

この原理図が図５－９に示されており，図中，ネットワーク要素の特別な成分であるスイッチは，分解して別々に示してある。スイッチは，蓄積交換機能をもつノードと同類である。この図は，ユーザ端末とノード及びセントロイド間のいくつかの可能な結合を示している。

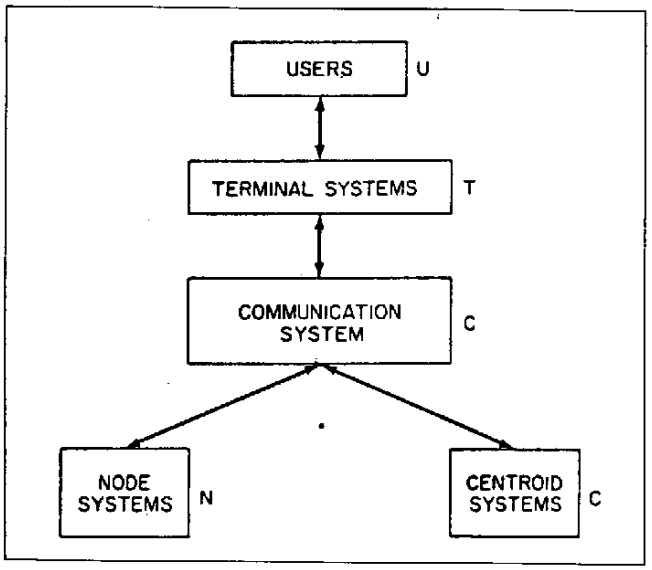


図５－８　Elements of Cybernet－Simple Relationships

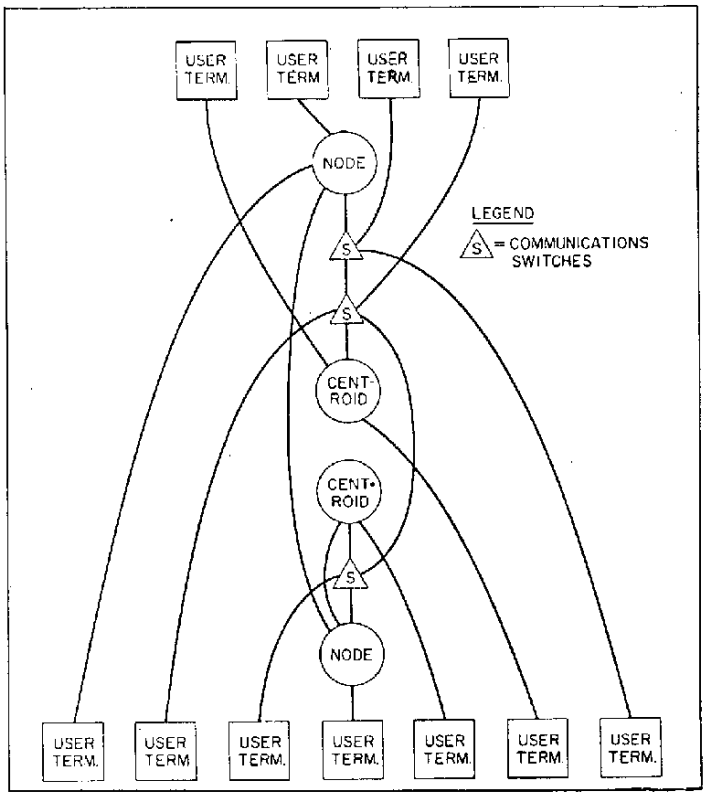


図5－9　Elements of Cybernet－Complex Relationships

C．DCSコンピュータ・ネットワーク

ＤＣＳ（Distributed Computer System）は，アービン（Irvine）にあるカルフォルニア大学で開発された実験的なコンピュータ・ネットワークである。ＤＣＳでは，低コストで，信頼性があり，新しいサービヌの追加が容易で，あまり高くないイニシャル・コストという点を開発目標にしている。初期の段階では，ミニとミディ・スケールのコンピュータをサービスすることを意図していた。その通信アーキテクチャは，基本的には，ベル・システムＴ１技術と固定長メッセージを利用するデジタル通信トポロジーに基ずいている。コンピュータは，リング・インタフェース（ring interface，コンピュータではない）と呼ばれる極めて高度なハードウエア装置を使用して，回路伝送媒体とインタフェースしている。通信プロトコル（protocol）上の特徴は，メッセージが，受信者がいる場所ではなく，受信者の名前によって受信者にアドレスされることである。

リング・インタフェースには，３つの役割りがあり，１つはフロント・エンド・マシンになりえるコンピュータをサポートし，１つは端末をリングへ直接取り付けでき，そして，もう１つは他のリングのネットワークを結合できるようになっている。この" ring of rings "は，基本リングと基本的に同じように運用する。

ＤＣＳは，分散型データ・ベースとユーザに対して各種のサービスを計画している。商業的に可能なシステムを作るというのでなく，むしろ，分布型ネットワークに含まれる問題点を探究しようとしている。しかし，実際に運用されるシステムは，そのアイデアをテストすることと，大学のキャンパスと保健センターにあるたくさんのミニコンの効率的利用を研究する目的で建設された。

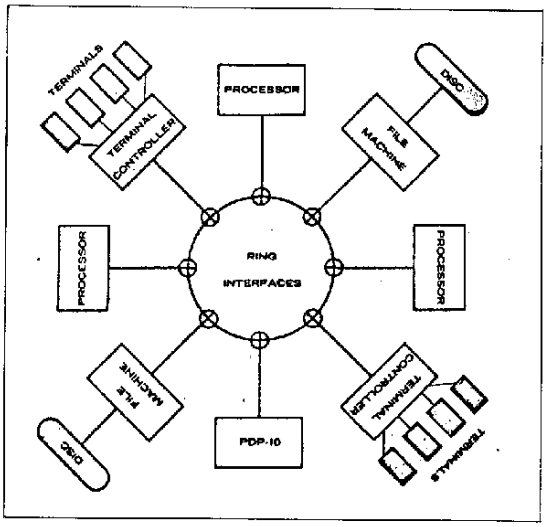


図５－10　The Distributed Computer System

D．MERITコンピュータ・ネットワーク

MERIT(Michigan Educational Research Information)コンピュータ・ネットワーク・プロジェクトの目的は，コンピュータ・リソース・シェアリングの研究である。1965 年，ミシガン州立大学委員会のグループによって，このプロジェクトの研究が着手された。1969 年までに，コンピュータ・ネットワークは，州議会，米国科学財団と３つの大学，すなわち，

ミシガン州立大学，ミシガン大学，そしてウェイン（Wayne）州立大学の協同作業によって作成された。

リソース・シェアリングに対する考え方は，ＡＲＰＡネットワークの目的と同じである。

ネットワークの構成は，各々のホストに接続した３つのノードの各々に通信制御機能を有するコンピュータ（ＤＥＣ社のＰＤＰ－11／20）を使用している。２つのノードにはＩＢＭ360／67，第３番目のノードにはＣＤＣ6500が，ホスト・コンピュータとして用いられている。

通信用コンピュータは，特別に設計したハードウエア・インタフェースを介してこれらのホストと接続している。各サイトを相互接続する通信回線は，2000bps の音声用回線である。

初期のインプリメンテーションでは，直通ダイアル，交換電話サービスを利用していた。その理由としては，１つには経済的理由，１つには低負荷という理由，もう１つは現在使用できるという理由である。しかし，結局，ネットワークの負荷が増大したので，他のもっと効率のよい経済的な設備を設置した。

通信用コンピュータ，ＰＤＰ－11／20は，ホスト・インタフェース・モジュール装置を介して，ＰＤＰ－11／20の主記憶装置からホスト・コアーと通信システムに可変長メッセージを転送する機能があり，ホスト・コンピュータが通信用コンピュータを周辺装置として扱うことが出来るようにしている。

通信用コンピュータ（ＣＣ）は，蓄積交換用システムとして稼動し，もし，経路が切断すれば，別のＣＣＳ経由で迂回するようなっている。

ＭＥＲＩＴの管理者は，コンピュータ・ネットワークは，全コンピュータ・エンバイロメントに利用上の相乗効果をもたらすと強く感じているが，一方，教育用コンピュータの相互接続に固有な管理面の困難さに直面している。

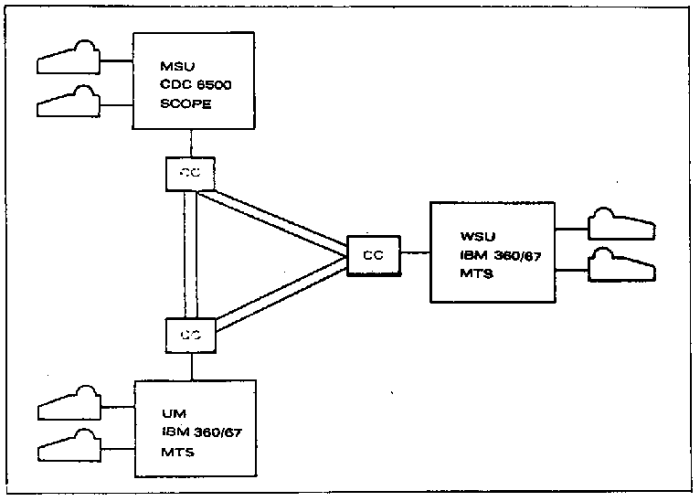


図5－11　MERIT LOGICAL MAP

CC：Communications Computer—DEC PDP-11.

E. NETWORK／440コンピュータ・ネットワーク

NETWORK／440 プロジェクトは，ネットワーク網についての研究を目的として，約5年前，ヨークタウン研究センター（Yorktown Research Center）で着手された。

NETWORK／440の論理的な構造は，図5-12に示してある。ネットワークは，2つの基本要素から成っている。ネットワーク・コントローラと通信サブシステム（Communication subsystem, CS）は，グリド・ノード（Grid Node, GN）と呼ばれるセントラル・システム中に常駐している。現在，これらの2つのモジュールは，OS／MVTの下で360／91のユーザ領域で走っている。第3番目のサブシステム，ユーザ・ノード・インタフェースは，ネットワークに参加しているリモート・ユーザ・システムに常駐している。

ネットワーク・コントローラは，ユーザが作成したネットワーク制御言語プログラムを解釈し，ジョブやデータをネットワークに送るため，適切なコマンドを発行すると同時に，ネットワーク中のすべてのユーザ・ノードのステータスを監視している。また，このコントローラにより，各ユーザのネットワーク・ジョブの同期実行も可能となっている。

加えて，NETWORK／440は，ユーザ・ノード（User Node, UN）でバッチ・ジョブの同時平行処理を行なうため，ネットワーク・ジョブ・キューを管理している。実行中のジョブは，ファイルとかデータの転送を開始するネットワーク・ジョブも含まれる。この時，現在アベイラブルでないユーザ・ノードへのメッセージ転送要求に対しては，記憶域を割当て，管理しておき，ノードが利用できるようになったら，これらのメッセージを転送する仕組になっている。

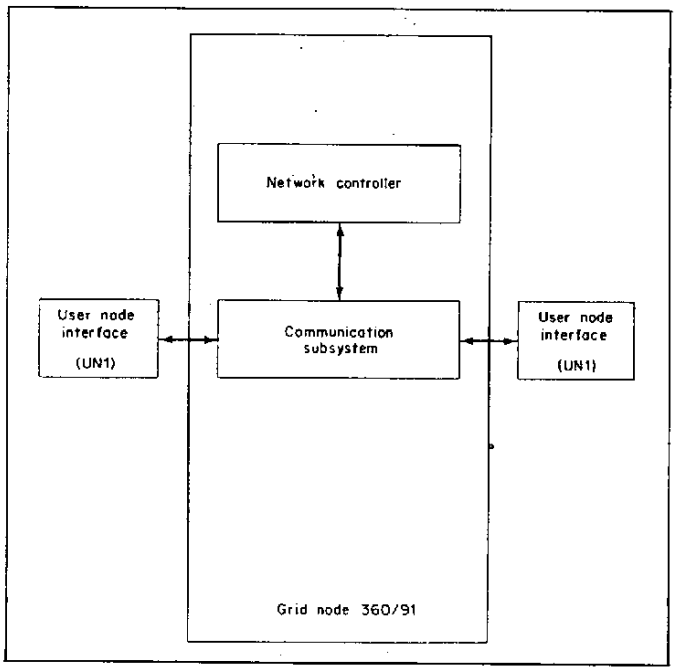


図5－12　Logical Structure of Network／440

通信サブシステム（ＣＳ）は，ネットワークを通過するすべてのメッセージに対して，センターとして動作する蓄積交換の機能があり，ネットワーク待ち行列中のすべてのメッセージを管理し，それらを適切な目的地に伝送する。また，ネットワーク・コントローラをユーザ・ノードとして取扱い，たとえ，それらが，同じマシンの中に両方とも常駐していても，同じ基本プロトコルに従う。

ＣＳは，ネットワークに対して，集合マルチプレクサとトラフィック・ディ

レクタとして振舞う。別の言葉でいうと，それは，通信回線から入ってくるトラフィックを受取り，適当な出力回線に送出したり，必要ならばメッセージを一時蓄積し，適切な出力シーケンスを確立する（図５－１３）。

論理メッセージを数個のブロックに分割し，ＧＮに伝送する。これらのブロックは，出て行く回線に物理的にマルチプレクスされる。ＣＳは，ネットワーク中のすべての回線の状態を監視し，オペレータ・コンソールに，この状態を記録し，ネットワーク・コントローラに状態の変化を通知する。

ＣＳプロトコルでは，ユーザ・ノードがあるノードへ，あるいは，あるノードからのマルチプレキシングの各種のレベルを指定することを許しており，もし，ある時刻に，ユーザ・ノードが，自分が要求していないメッセージを受取ったならば，それを棄却し，後で，そのメッセージの伝送を要求することができる。ＣＳは，ユーザ・ノードがメッセージを受信できるようになるまで，その論理メッセージのすべてのブロックを貯えることができる。加えて，ユーザ・ノードは，それに，＂no new bussiness＂メッセージを伝送することを要求できる。すべての新しいトランザクションは，再開メッセ

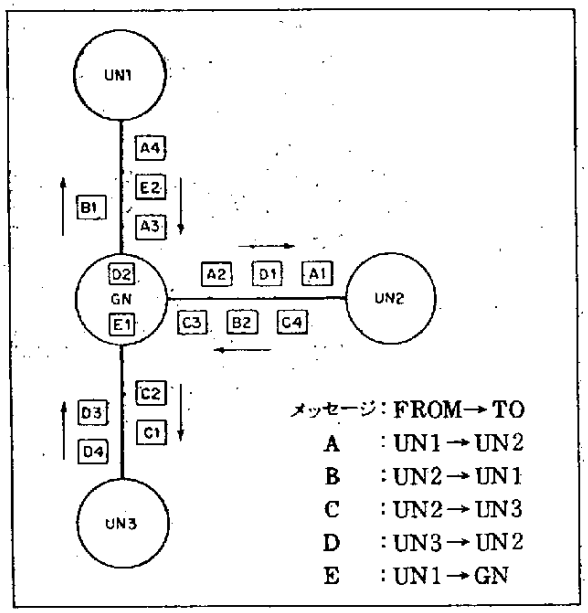


図５－１３　Ｌｉｎｅ Ｍｕｌｔｉｐｌｅｘｉｎｇ

ージが，目的地ユーザ・ノードによって送出されるまで，ＣＳ中に貯えられる。

グリン・ノードに入ってきたり，出て行くすべてのメッセージの時系列の記録が，ＣＳによって作成される。それは，ＩＢＭのＢＴＡＭエラー・リカバリ処理よりももっと拡張されたエラー・リカバリ処理を行なっている。

ＣＳは，次の通信規約を使って稼動している。

(1) コンテンション・ノードとの同期伝送通信。

(2) 2つの端末で同時に要求が起った場合は，優先権処理を行なう。

(3) 2.4 ～ 40.8 キロビットの帯域帯をもつ賃借半2重回線。

最後に，ネットワーク運用に関して，全二重オペレーション，フロントーエンド・マシン・オペレーション，前進エラー修正機能，そして，最適ネットワーク設計の一部としてネットワーク・アクセス・メソッドが研究されている。

図5－14に，現在の運用形態を示す。

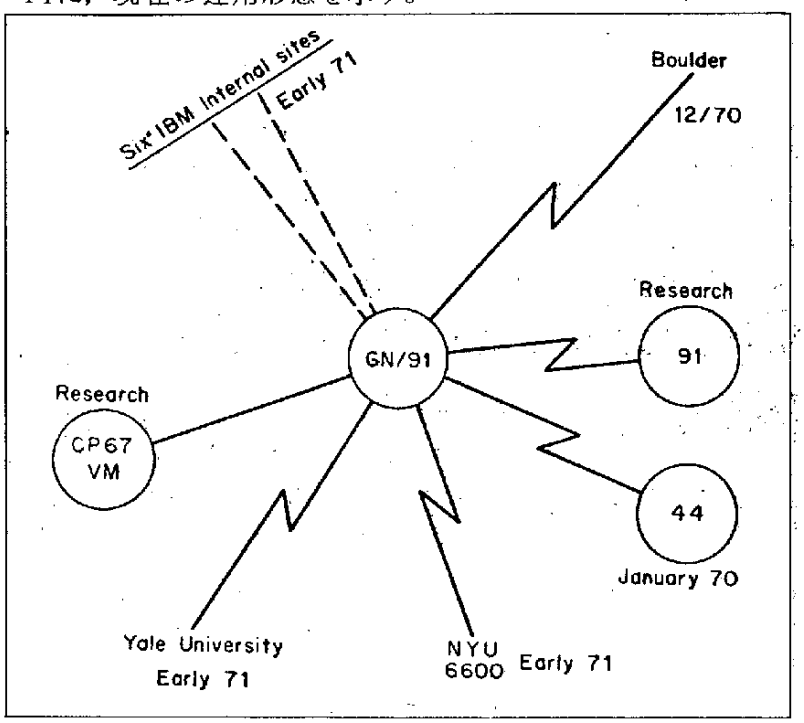


図5－14　NETWORK／440 LOGICAL MAP

F．NPLコンピュータ・ネットワーク

英国の国立物理研究所（National Pysical Laboratory，ＮＰＬ）のＮＰＬネットワークは，英国のNational Networkの実現を前提として，試験的，実験的要素を含め，現実の強い要求にともどいて，この研究所内に設置されたネットワークである。このネットワークは，将来のNational Networkのローカル・ネットワークとなる可能性がある。

ＮＰＬにおけるLaboratory Networkの要求は，次のような諸点がある。

① 簡単な身近な端末から中央のコンピュータへ容易にアクセス出来ること。特にプログラムの作成，修正，デバッグ，テストラン等が即座に自分の研究室からおこなえること。

② 計算結果が出来るだけ早いターンアラウンドで見られること。また結果が膨大で大量のプリントアウトやグラフ出力の要求が多いのでそれらの装置が至る所に設置されることが望ましい。

③ インタラクティブなグラフィック装置が多くの研究で使われているが，そのためのローカル・コンピュータは機能の制約が多く，より大きなリソースにアクセス出来るとより有効である。

④ 研究室単位に設置されているローカル・コンピュータでも，プログラムやデータの量が多くなると，適当な２次メモリーが必要となる。これらのメモリーは方々に分散するよりも集中化した方が経済的であり，且つより大きなエリアを確保できる可能性もある。

⑤ ローカル・コンピュータでは不可能な大規模の計算も，それを通して自動的により大きなリソースにアクセスされ，目的に合ったリソースを効率よく使用することが出来る。

⑥ 外部のコンピュータ・サービスあるいはネットワークと電話回線によってコネクトすることによって，サービスの利用の範囲が広がる。

⑦ コンピュータ利用に関する一切の管理をネットワークを通しておこなうことが出来る。

現在，研究所内の27個所が直接，間接に１メガビット／秒のライン（電話

回線ではない）でコネクトされている。このうち１つは約４マイル離れたShip Divisionの研究室にもコネクトされている（250キロビット／秒）。

このネットワークは1970年7月から運転が開始され，現在も規模を拡張中であるという。

制御方法は図5－15および5－16に示すように多段のMultiplexerが夫々最大8個つづPCU(Peripheral Control Unit)から入って来る8ビットづつのパラレル・ビットをアキュムレートし，MSC(Message Switching Computer)で1024ビット（128バイト）のパケット(Packet)に組み立てられる。

この手法は，いろいろのことなるペリフェラル装置，例えば実験室や研究室にある中小のコンピュータ自身をはじめ，テレタイプ，キャラクター・ディスプレイ，グラフィック・ディスプレイ，プロッタ，データ・ロガー(Data logger),磁気テープ，カード・リーダ，ライン・プリンター等それぞれ全く異なる速度の装置のすべてに適用可能であり，更にネットワークの拡張性

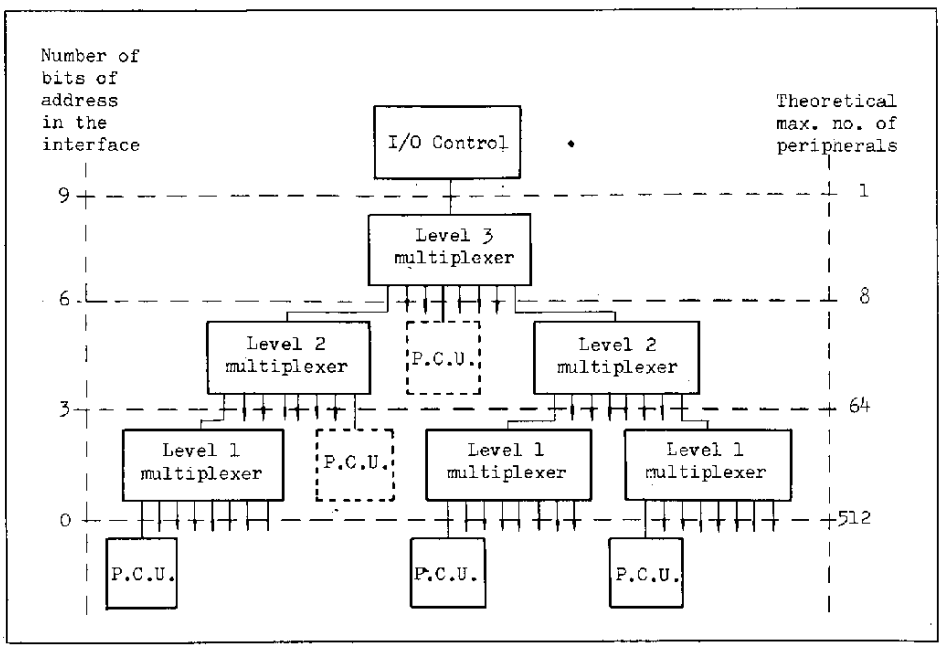


図5－15　マルチプレクサーによるコントロールの概念図
(PCU:Peripheral Control Unit)

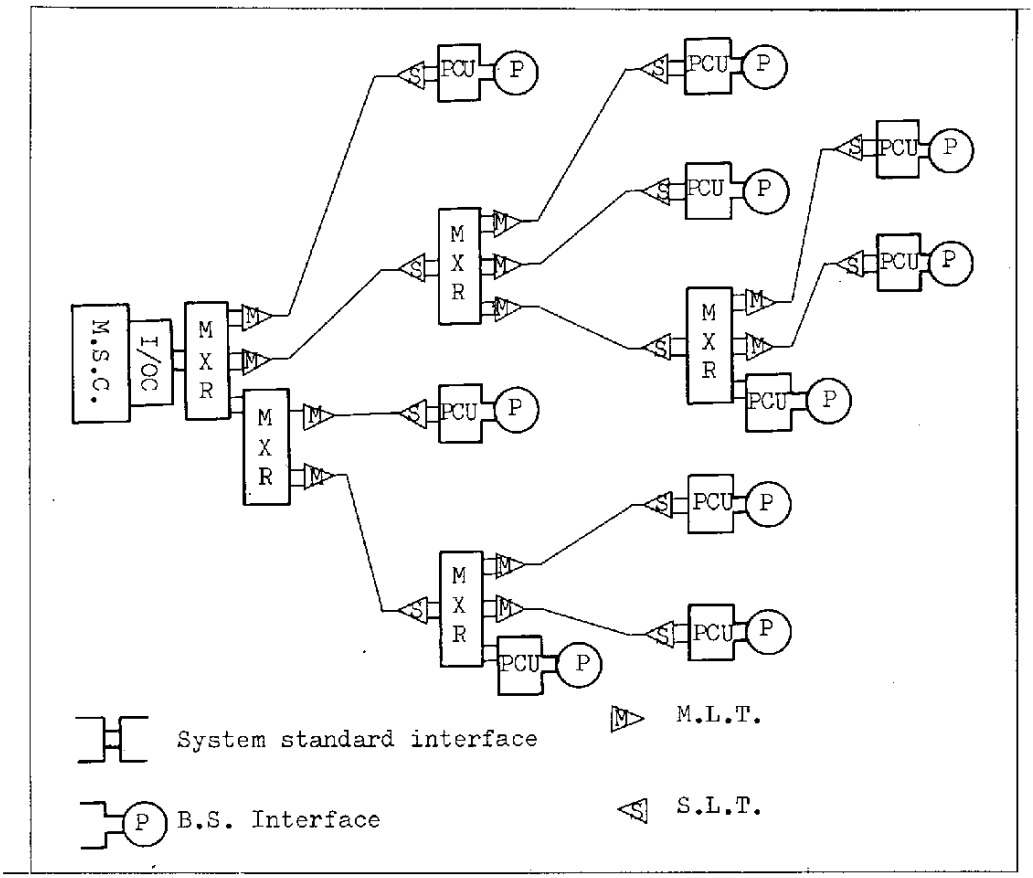


図5－16　NPLネットワークのコントロール構成図

も自由である。各種の装置間のインタフェースには British Standard Interface が適用されている。

セントラル・コンピュータとみなされるのはＫＤＦ９と呼ぶＩＣＬの古いコンピュータであるが，その他ＥＬＬＩＯＴＴ４１２０，Ｈ３１６，ＭＩＣＲＯ－１６，ＭＯＤＵＬＥＲ－Ｉ，ＰＤＰ８×３台および大容量ディスク等がコネクトされているＭＳＣとしては現在ＤＤＰ－５１６を使用している。

このネットワークの使用形態は，インタラクティブな使用とファイル転送のような比較的転送量の多いものとがある。このネットワークの設計条件の一つは両方の目的にオプティマムなものとすることである。インタラクティブなものは１キャラクター単位の処理でよいが，後者の場合はパケットのブロッキングの機能によって転送の効率をあげている。また前者の場合はレス

ポンス・タイムが問題になるが，端末対端末の応答は殆んど気にならないレスポンス・タイムである。但し，インタラクティブなピクチュアーの転送は，この速度では無理だという。少くとも128バイトの単位では一画面の絵にはならない。

G. OCTOPUSコンピュータ・ネットワーク

OCTOPUSシステムは，ローレンス・バークレイ研究所（Lawrence Berkeley Laboratory）（以前のローレンス放射線研究所）で開発された異機種コンピュータ・ネットワークである。このシステムは，2台のCDC 6600と2台のCDC7600と，将来，CDC STARを結合している（図5－17参照）。これらのホストは，ワーカズ（workers）と呼ばれ，すべてタイム・シェアリング・システムとして稼動している。研究所としては，集中階層データ・ベースとネットワークを単一リースと見做すことのできるバラエティに富んだ入出力装置の設置を計画している。

通信システムは，蓄積交換方式プロトコルを採用している。OCTOPUSのワーカズは，12メガビットの容量のケーブルで相互結合されている。システムは，各々，特別の役割りをもつ2つのサブネットワークと考えることができる。第1番目は，ワーカズ，伝送制御コンピュータ，デュアルDECシステムとファイル記憶装置からなるファイル伝送サブネットである（図5－18参照）。第2番目のネットワークは，PDP－8（各々は128個の端末をサポートしている），ワーカズ，そして，伝送制御コンピュータからなるテレタイプ・ネットワークである（図5－19参照）。テレタイプ・サブネットワークは分布型ネットワークで，一方，ファイル伝送サブネットワークは集中型サブネットワークである。デュアルDECシステム10は，この集中型サブネットの信頼度を高める。

OCTOPUSは，現在，稼動中のもっとも精巧なネットワークの1つであり，ファイルの機密保護にも十分考慮を加えて設計された数少ないネットワークの1つでもある。

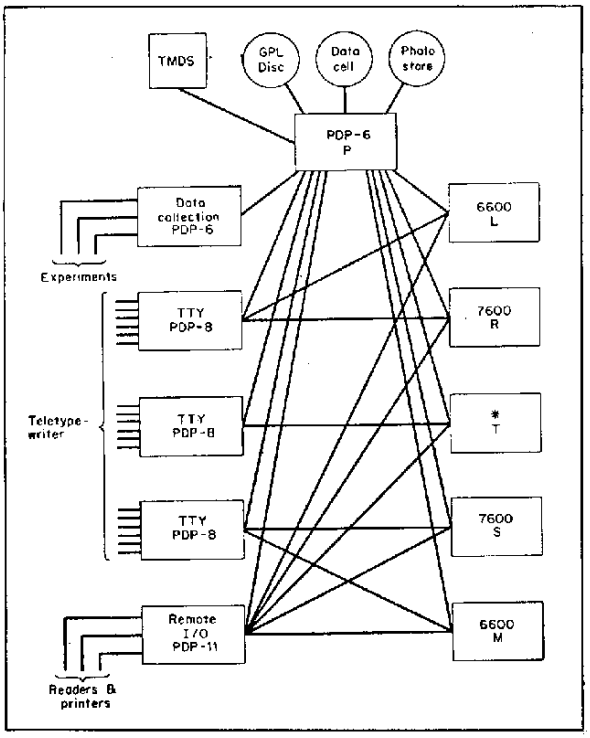


図5－17　Octopus Network Fall 1970

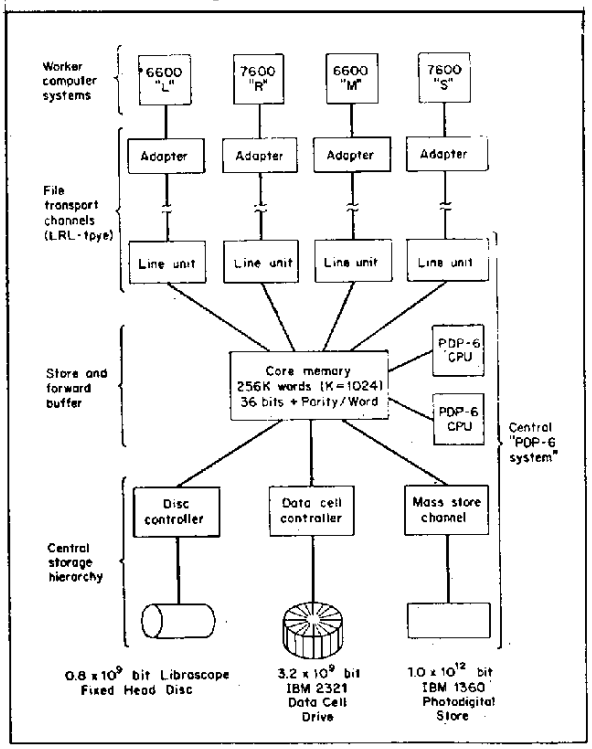


図4－18　Octopus File Transport Network

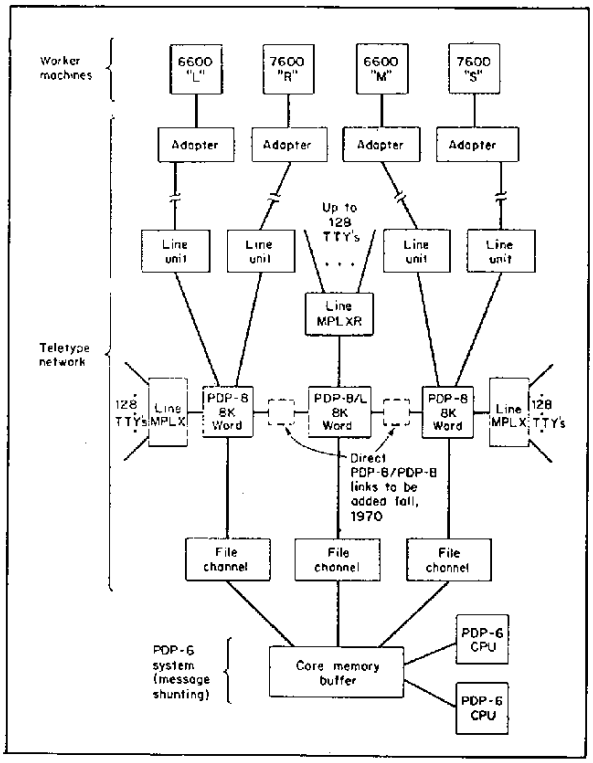


図5－19　Octopus Teletype Network

H．TSSコンピュータ・ネットワーク

ＴＳＳネットワーク（ＩＢＭと360／67カストマーの一部の共同事業）は，分布型で稼動する同機種コンピュータのネットワークとして開発された。ＴＳＳネットワークで稼動しているホストの各々は，ＩＢＭ TSS／360オペレーティング・システムを使っている360／67である。

360／67間を結ぶ通信装置は，音声用交換回線を利用している。これらの回線は，ＩＢＭ2701,又は，2703を使用して360／67とインタフェースしている。この様に，このネットワークは，標準的なハードウエアを利用しているので，通信プロトコルに関しては，プログラムを組む必要がない。この様に，蓄積交換とかエラー制御等のすべてのプログラムは，ホスト・マシンに常駐している。事実，通信用ソフトウエアは，アクセス・メソッドを介してユーザ・プログラムとして動く。将来は，ＩＢＭ370／145を通信用コン

ピュータとデータ・ベース管理用として動かす計画がある。また，需要があれば，50,000bpsの回線を利用する計画がある。

ＴＳＳネットワークは，実験的なものである。ネットワーク中のすべてのマシンは，同機種であるので，プログラムとデータの相互交換が容易にできる。このネットワークでは，ダイナミック・ファイル・アクセスとリモート・バッチの両方が利用できる。

Ｉ．TUCCコンピュータ・ネットワーク

ＴＵＣＣ（The Triangle Universities Computation Center）は，ノース・カロライナ州にある３つの主要大学，すなわち，ダーハム(Durham)にあるデューク（Duke）大学，チャペル・ヒル（Chapel Hill）にあるノース・カロライナ（North Corolina）大学とレイリー(Raleigh)にあるノース・カロライナ州立大学の共同事業として，1965 年に設立された。

最初の動機は，経済的理由である。すなわち，コンピュータを個々のサイトに置くよりも低いコストでより多くのコンピュータ・パワーを各々の大学がアクセスできるようにすることである。

ＴＵＣＣは，中央サービス・ノード（ＩＢＭ370／165）と以下に示す３レベル・ジョブ・ソース・ノード群からなる中央集中型の同機種（homogeneous）ネットワークである。

(1) １次ジョブ・ソース・ノード３個
- ＩＢＭ360／75，ＩＢＭ360／40，ＩＢＭ360／40
- センタ入力ジョブを扱う

(2) ２次ジョブ・ソース・ノード23個
- 専用回線用Data 100，UCC1200，IBM1130，IBM2780と専用回線，又は，電話回線用ＩBM2770
- リモート・ステーション入力ジョブを扱う。

(3) ３次ジョブ・ソース・ノード125個

・64個の電話，又は，専用回線用Teletype 33 ASR，IBM1050，IBM2741，UCC1035，その他

・端末入力ジョブを扱う。

ネットのノードは，賃借の40,800bpsの半2重回線で中央ノードのコンピュータに接続している。この回線は，IBM2701 データ・アダプタを用いて360システムにインタフェースしている。ネット中のすべてのソース・ノード・コンピュータは，中央サービス・ノードと同機種である。そして遠隔処理サービスに加えてローカル計算サービスを行なっているけど，現在，ノンローカル（non-local）なネットワーク計算サービスは行なっていない。しかし，1次ソース・ノードでのネットワーク計算サービスを行なう技術は，他のレベルでもそのまま利用でき，この技術を使ういくつかの重要な計画がある。

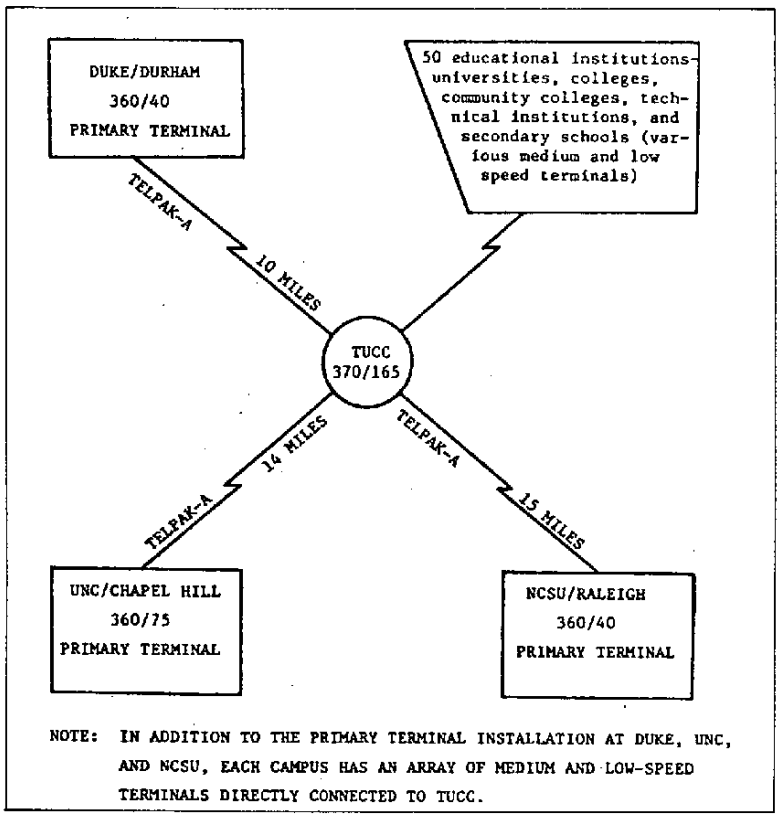


図5－20　The TUCC network

現在，ＴＵＣＣは，各種の伝送速度の通信回線でセンタと州内の50の小さい公共機関と２・３の研究所にあるコンピュータ端末施設とを接続している。

ＴＵＣＣは，　S／３６０－MVT／HASPを使って，２メガバイトの記憶容量をもつ通信用のＩＢＭ370／165 の元で稼動しており，多種類の端末をサポートしている（図５－22参照）。チャペル・ヒルにある360／75，ノース・カロライナ州とデュークにある360／40は高速通信回線の結ばれている。３つのキャンパスにある計算センタは完全に同機種である。ＴＵＣＣは，見方を変えると，ユーザにサービスするために大規模な付加的な計算力をもったパイプライン方式のコンピュータと考えられる。

次に，ＴＵＣＣユーザ・コミィティに対するサービスは，リモート・ジョブ・エントリ（ＲＪＥ）とインタラクティブ処理の両方である。インタラクティブ・サービスのプログラミング・システムとしては，BASIC，PL／I，とＡＰＬ言語が利用可能である。又，ＴＳＯ（Time Sharing Options）は，実験的モードでランしている。ＲＪＥ（Remote Job Entry）を通して利用できるものは，FORTRAN Ⅳ，PL／1，COBOL，ALGOL，PL／C，WATFIV　と　WATBOLの大規模のコンパイラ群である。これらの言語ファシリティは，学術計算用の広範囲のアプリケーション・ライブラリに結合している。

他にも，重要な利点がある。第１番目は，広範囲のアプリケーション・プログラムの共同利用である。プログラムは，１つの団体で開発されると，それをネットワーク中のどこからでも容易に使用することができる。所有権のあるプログラムの場合は，通常，僅かではあるが料金を支払う必要がある。

第２番目は，研究や教育のために，大規模な計算を必要とする学部のメンバと大学の計算機科学部門に属している人を含めたＴＵＣＣのスタッフ・メンバがこのシステムを利用できる。

第３番目の利点は，非常に高度な能力をもったシステム・プログラマとか管理者が少ないので，センタに集中的に集めることができる。

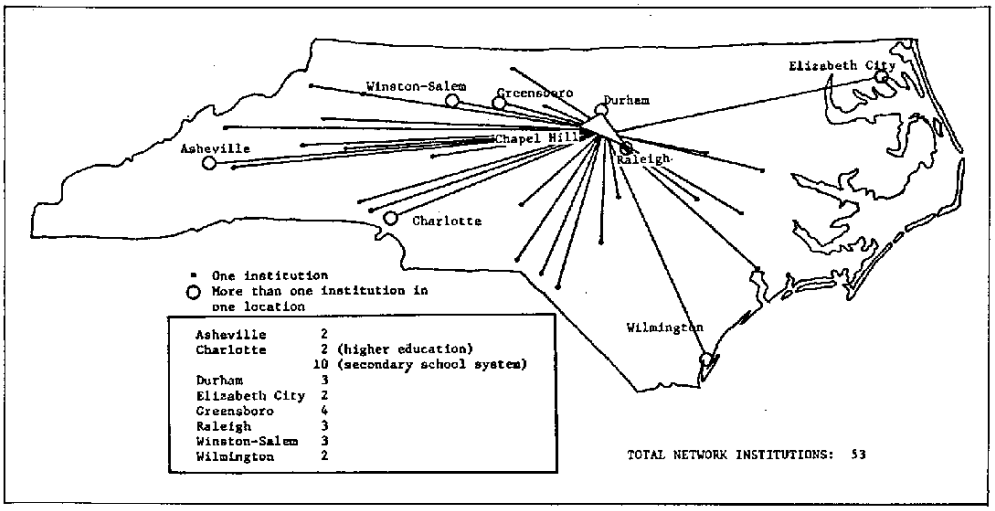


図5－21　Network of institutions served by TUCC/NCECD

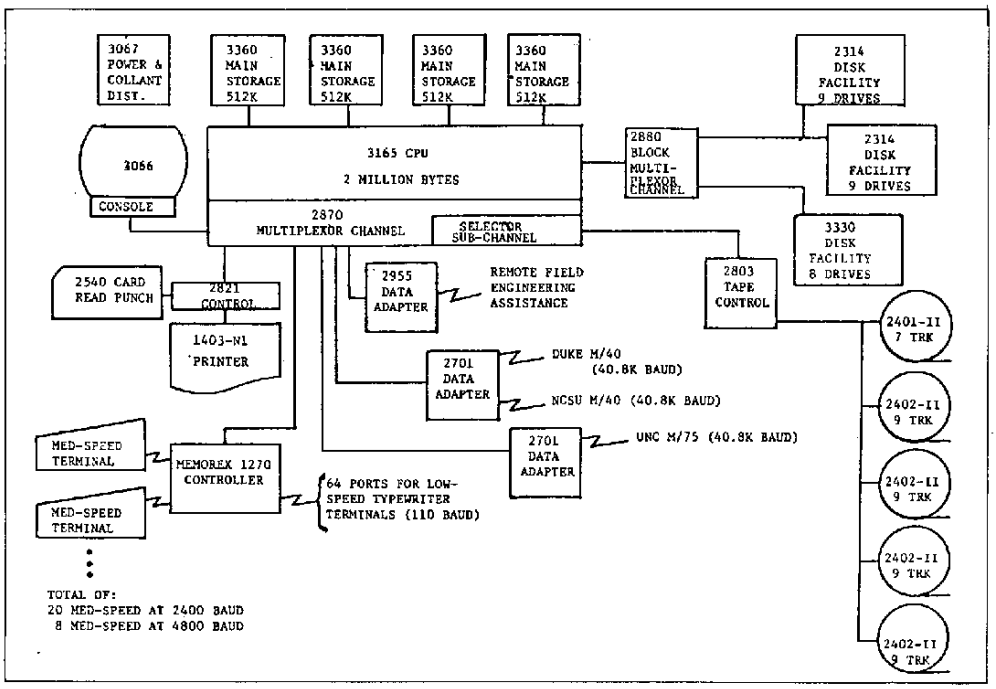


図5－22　TUCC hardware configuration

ＴＵＣＣは，ユーザに対してcost-effectivs な一般計算サービスを続けている。予想される改良点としては，

- ＴＳＯを経由して利用される広範囲なインタラクティブ・サービス
- ＴＵＣＣで開発した対称ＨＡＳＰ－ｔｏ－ＨＡＳＰ ソフトウエアの手法によってネットワークを多重サービス・ノード・ネットワークに発展させる。
- 低速の端末に対するメッセージの集中機能を有する中速の端末の準備，このことは通信コストを最小にする。
- 通信コストを減小させるライン・マルチプレクサの使用。
- 広範囲のデータ処理に対する端末サービスの拡張。

である。

5.2 コンピュータ・ネットワーク設計技術と問題点

（ARPANETを中心に）

コンピュータ・ネットワークの設計目標は，スループット，遅延時間，信頼性そしてコストを最適化することである。

ネットワークを設計する場合に利用できる原理的な道具は，解析，シミュレーション，ヒューリステック・プロセデュア（heuristic procedure），そして実験である。解析，シミュレーションとヒューリステックは，モデリング（modeling）とトポロジカル（topological）の最適化の研究の中心であり，一方，シミュレーション，ヒューリステック・プロセデュアと実験技術は，実際のネットワーク・インプリメンテーションに対する主要な道具である。これらのすべてが万能というのではなく，これらの方法のすべては役に立つ。もっとも有効的なアプローチは，これらの道具のいくつかを同時に使うことである。

各々のアプローチは，かなり改良の余地がある。解析研究は，経路選択のような重要な面での成果は，まだ限られている。しかし，遅延時間の予測の場合には，簡単な閾値モデルを導き，それは，正確で理解しやすい。ヒューリステック・プロセデュアでの大規模なネットワークに対する適切なヒューリステックをどのようにして選ぶかという問題がまだ未解決である。数百以下の交換機を含むネットワークを設計する場合は，現在のヒューリステックでほぼ十分であり，しかし，より規模の大きいネットワークに対しては，さらに研究する必要がある。

シミュレーションの適用は，一般に，費用がかかり，シミュレーション・モデルの正当性とその結果を完全に分析することは，かなりむずかしい問題であるが，しばしば利用されるすぐれた道具の１つである。シミュレーション・モデルを検証したり，交換機が持たなければならないバッファ数のような詳細な設計の決定を助けたりするのにもっとも役に立つように思われる。

しかし，ネットワークのサイズが大きくなるにつれて，モデルの正当性の評価がむずかしく，かつ，費用がかさむので，シミュレーションは，トータルの

設計器具として，事実上役に立たなくなってくる。すべてのモデルや結論をテストする最後の道具は，実験である。実際のネットワークでの実験は，非常に有効である。もちろん，実施にともなう種々の困難はあるが，これは，実験そのものの効果と同時に，モデルやヒューリステックの正当性のチェック，あるいは，設計パラメータのテストなどにも有効な手法である。

この節では，ARPANETで使われた設計技術をレビューし，評価する。

まず，IMP設計，トポロジカル設計について述べ，更に，ネットワーク・モデルに関する主な問題点および設計技術についても紹介する。

ARPAネットでの主な設計項目は，次の4点である。

(1) ノードで蓄積交換として働くIMPの設計。

(2) ネットワーク中の各々の通信回線の容量や配置をきめるためのトポロジカル設計。

(3) タイム・シェアリング，バッチ処理，そして他のデータ処理システムによるネットワークの使用に対する高レベル・プロトコルの設計。

(4) ネットワーク・パフォーマンスのモデリングと測定。

5.2.1 IMPの設計

IMPの設計は，大きく2つに分けられる。

- ハードウエアの構成（タイミングと信頼性と運用管理プロセデュアを基本にして）
- 設計したIMPハードウエアを使用するオペレーテング・プロセデュアの設計とインプリメンテーション

ARPANETのために初期に開発されたIMPは，ほぼ，1メガビット／秒の回線容量（すなわち，50キロビット／秒の伝送速度の回線が20個）があり，有効情報を約3／4メガビット／秒で処理できる，16ビット語長，サイクル・タイム1マイクロ秒のコンピュータである。ハードウエアは，必要なIMP容量によって変化するであろう。しかし，1つのIMP容量でうまく働くオペレーティング・プロセデュアは，異った容量のマシンにも多分，移行することがで

きる。しかし，ネットワークの規模や利用が拡大すると，階層的なトポロジーのような構造的特性を考慮したもっと機能の大きいオペレーティング・プロセデュアが必要となる。

ＩＭＰ設計の４つの基本的な事項，メッセージ処理とバッファ管理，エラー制御，フロー制御，そして経路選択につき検討する。ＩＭＰは，ホストからのメッセージや隣接するＩＭＰからきたパケットを処理するためのバッファをもっている。エラー制御は，雑音のある通信回線でホスト・メッセージを高信頼度で通信をするために必要である。オペレーティング・プロセデュアの設計目標は，重トラフィック負荷の下で，高い実効スループットを実現することである。この目標を遂行するための潜在的な障害は，次の２点である。

(1) 負荷が増加するにつれて，ネットは，輻輳（ふくそう，congest）状態になり，スループットを減少させる。

(2) 経路選択プロセデュアは，ネットワーク全体の効率的な経路選択を保証するためのもので，パケット自身がいつも高速に移動するとはかぎらない。

フロー制御と経路選択プロセデュアは，この要求に十分応じられなければならない。

Ａ．メッセージ処理とバッファ管理

ARPANETにおいて，最高メッセージ長は，約8000 ビット以下に制限されている。一般に，１組のホストは，一連の伝送メッセージをネットを経由して通信する。そのようなメッセージに対して遅延時間を数十秒におさえ，必要なＩＭＰバッファ記憶域を減少するために，ＩＭＰプログラムは，各々のメッセージをほぼ1000 ビット位の１つ以上のパケットに分解する。メッセージの各々のパケットは，目的地ＩＭＰに向けて独立に伝送され，目的地ＩＭＰで，その目的地ホストへ送る前に，ＩＭＰプログラムによって，メッセージに再組立てする。一方，ホストは，再組立てされたメッセージを処理し，応答を返す。非同期のＩＭＰ－ホスト・チャンネルの場合，これは，ＩＭＰタスクをわずかに簡単にする。しかし，すべてのＩＭＰ－ホスト・チャ

は個々のパケットの動きに焦点を合わすのではなく，むしろ，経路を選択するとき，トポロジカルや統計情報を使う。例えば，平均化プロセデュアを使用することによって，トラフィック・フローを平滑化することができる。トラフィックを平滑化する利点は，経路選択アルゴリズムが，各ＩＭＰの経路選択表を調節するための時間が十分にとれることである。

ＡＲＰＡネットで初期に使われた方式は，各ＩＭＰが，目的地に対する出線の内，目的地への最小遅延時間であると推定された回線が選択され，その経路に沿ってパケットを運んでいる。

選択表は，隣接するＩＭＰから最小時間推定値を教えてもらって，隣接するＩＭＰへの遅延時間の内部的推定値を使って０.５秒毎に更新された。たとえ，経路選択アルゴリズムが，目的地当り，１回に１回線だけを選択するとしても，もし，経路選択表が更新され，別の回線が選択される前に，１つの回線上を伝送されるのを待っているパケットの待ち行列が組立てられていたならば，２個の出線が使用される。（経路選択が変更されない間の）更新間隔（その間は，経路選択は変らない）中で使用される目的地当り，数個の回線の使用を許す変形された方式では，経路を選択するために２つ以上の遅延時間推定値を使うことは可能である。

実際上，このアプローチは，ネット中で経験した中レベルのトラフィックで，まったく効果的に作動した。重トラフィック・フローでは，この方式は，変化に関する情報が伝達されるよりも早く変更ができるようになっているが，経路選択情報は，待ち行列の長さにもとずいているので効果的でない。この変化に関する情報は，事実上まにあわず，一般に，重トラフィック・ケースでは，効果的な経路選択を決定するときの助けとならないが，幸運にも，この情報はまだ経路選択に利用できる。

重トラフィックの間だけ効果的に使用するため多重経路を許すもっと複雑な方式が，BBN(Bolt Beranek and Newman Inc.)で最近開発された。

予備シミュレーション実験は，それが重トラフィック情況の変化に対してネットワーク中の効果的な経路選択ができるように作られていることを示し

た。この方式は，特別なパケットが通る経路の選択問題と普通のパケットが通る経路の決定問題を別々に取扱っている。この方式によって，もっとも少ない個数のＩＭＰを経由する経路でパケットを運ぶことが可能である。しかし，その経路の容量を越えていれば，すなわち，その経路が一杯の時，次に少ない個数のＩＭＰを経由する経路でパケットを運ぶ。以下同様に順次くり返される。経路選択に関する同様なアプローチは，経路選択の決定にＩＭＰが関与する必要のない理想化された方式で，ＮＡＣ（Network Analysis Center）によって，独立に開発された。また，フロー偏差技法を使った別のアプローチが，最近，ＵＣＬＡで研究中である。このアプローチにもとずいて作成された測定機能をもったプロセデュアは，定常状態では，全体のネットワーク・フローをゆっくり変化させて，フローを増加させるために現在のフロー・パターンを乱すことを行なっている。これらのアプローチは，望ましい経路選択方式に共通な基本要素をもっている。

5.2.2　トポロジカルな考察

トポロジカル設計とは，与えられたコストに対して高スループットを供給する効果的なネットワーク・トポロジーを得ることである。スループット率には，たくさんの測定基準があるが，ここでは，すべてのＩＭＰが，きめられたトラフィック・パターンに従ってトラフィックを伝送しているとき，１個のＩＭＰが，ネットワークに送出できる平均トラフィック量ときめる。一般には，すべてのＩＭＰが，まったく同じように振舞い，かつ，各ＩＭＰは，隣接するＩＭＰへ同じトラフィック量を送出すると仮定される。トポロジカル設計に含まれる要素として，

(1)　利用可能な回線（Common carrier circuits）

(2)　目標額またはスループット

(3)　信頼性

(4)　設計計算費用

などである。

ようにする必要がある。余分のバッファ記憶域を備えておけば，輻輳やデッドロックを緩和するが，しかし，一般に，それらを防止することはできない。

目的地ＩＭＰから到着率でパケットを受取る目的地ホストの持続障害は，ネットワークを充満させ，輻輳状態にさせる原因となる。又，再組立て閉鎖と蓄積交換閉鎖として知られている２種類の論理的なデッドロックが起るかもしれない。再組立て閉鎖というのは，部分的に再組立てされたメッセージの残りのパケットが，その目的地の再組立て用記憶域があくのを待っているネット中のパケットによって，目的地ＩＭＰに到着するのを閉塞されている（メッセージが完了した後で，再組立て用記憶域をフリーするので）場合を言う。蓄積交換閉鎖というのは，目的地ＩＭＰには，到着するパケットを受付けるための空地はあるが，通過経路中のバッファがないため，相互干渉し，その結果として，どのパケットも目的地に到着することができなくなる場合をいう。これらの現象は，ＡＲＰＡＮＥＴの非常に綿密に計画されたテスト期間中に起り，又，シミュレーションでも起っている。

初期のＡＲＰＡＮＥＴの設計では，ソフトウエア・リンクとＲＦＮＭ信号の使用は，１つのリンク，又は，リンクの小集合によって輻輳を防止していた。しかし，１つのＩＭＰにたくさんのリンクが集まってしまうとトラフィックによって，また，輻輳を起してしまう。この機構は，閉鎖を防止できないけれど，その手法は，現在まで，ネットで遭偶したトラフィック・レベルに対しては，十分な保護機構を提供する。輻輳と閉鎖を防止するための初期のＩＭＰ設計に使われた特に簡単なフロー制御アルゴリズムは，輻輳がまさに起ろうとしたとき，目的地ＩＭＰでネットからきたパケットを捨て，発信地ホストのＩＭＰで，すこし遅れて各々の捨てられたパケットの写しを再送する機構となっている。トラフィック・レベルが高くなると，遅延時間が実験値よりもかなり増加するので，ネット中の伝送遅延時間が小さな値をもちつづけるように，トラフィックをネット以外の場所の待ち行列に入れる。インタラクティブ・トラフィックに対しては短い遅延時間で，ファイルの高速伝送

に対しては広帯域という２つの要求がある。広帯域伝送を許すために，ネットは，１つのホストからパケットの定常流（steady flow）を受取り，同時に，目的地で切迫した輻輳が起ったとき，発信地ＩＭＰへの入口でフローをすばやく消すことができなければならない。これは，普通，短いインタラクティブ・メッセージを不必要に長い遅れをこうもることから保護するために，アルゴリズム中に特に組込まれる必要がある。

D．経路選択方式

分布型ネットワークに対するネットワークの経路選択で利用できる情報のみで経路選択を行なう経路選択方式がＩＭＰ内に必要となる。そのような方式の簡単な例は，各々のＩＭＰが，目的地への最短経路として現在推定した経路に沿って，パケットを独立に送出する方式である。

ＩＭＰは，他のＩＭＰから受取った情報と，回線網のalive／dead 状態のような局所情報を使って経路選択の計算を実行する。普通，負荷が刻々かわるので，効率的な経路選択を行なうには，内部待ち行列の長さのような局所情報だけでは不十分で，隣接するＩＭＰからの支援を必要とする。各々のＩＭＰにネットワークのトポロジカル・マップを必要とせずに効率的な経路選択を行なうために，ＩＭＰに小量の計算能力を持たし，隣接するＩＭＰから効率的な情報を得ることは可能である。トラフィック・パターンが規則性を示すようなアプリケーションにおいて，中央コントローラの使用が望ましい。しかし，ダイナミックにトラフィックが変わるようなアプリケーションにおいては，中央コントローラが，むりに，ネットを経由してその情報を伝達したならば，ＩＭＰが経路選択表を更新する以上には，中央コントローラを効果的に使うことはできないように思われる。また，経路選択の決定機構をＩＭＰの間に分散することよりも信頼できない経路選択のアプローチである。

経路選択情報の収集と伝達および経路割当て計算に時間がかかり，経路選択情報は，瞬時に変化するトラフィック・フローを正確に反映するに必要な時間内でネットに伝播することはできない。それ故に，効率的アルゴリズム

ンネルが同期をとり，ホストが再組立て処理を行なうならば，ＩＭＰタスクは，さらに簡単になる。この後者の場合，ＩＭＰにあるプログラムとほぼ同じ機能のソフトウエアは，すべてのホストの中になければいけない。

バッファ処理の方法は，処理を高速で行ない，プログラムを小さくするため，当然，簡単にしなければならない。

使用する回線の容量に応じたたくさんのパケットを格納するに十分なバッファの個数が必要である。バッファ・サイズは，簡単な解析手法できめられる。例えば，ＩＭＰにおける固定長バッファの利用は，設計を簡素化し，運用速度を早めることができる。しかし，可変長パケットの場合，当然利用率がさがる。もし，各々のバッファが，Ａ語のオーバヘッドと，Ｍ語の本文を含むものとし，もし，メッセージ長が１からＬまで一様分析をしているとするとすれば，必要とする記憶域を最小とするＭの値は，ほぼ$\sqrt{\mathrm{AL}}$であること示すことができる。実際上，ほとんどのトラフィックは短く，オーバヘッド量は，バッファ記憶域の25％位であるので，Ｍの値は，やや小さくしてある。

Ｂ．エラー制御

ＩＭＰは，エラー制御を行なわなければならない。可能性のあるエラーとして，次の４つがある。

(1) メッセージは，目的地へ運ばれる順番がくるってしまう。

(2) 重複メッセージが，目的地へ向けて運ばれる。

(3) 伝送中に，メッセージにエラーが入り込む。

(4) メッセージが，運ばれない。

目的地ホストに向けて運ばれるメッセージの正しいシーケンス制御を行なうタスクは，実際にはエラー制御とフロー制御の両方の職分に入る。もし，ホスト間で１度には１つのメッセージしかネット中に入ることを許さないならば，正しいシーケンス制御は簡単である。

ＡＣＫ処理をミスした後で，重複パケットは，目的地ＩＭＰに到着する。このことは，連続的に受信されたパケットを再送する原因になる。ＩＭＰは，

パケットがネットワークに入ってきたとき，各々のパケットに一連番号を割当て，目的地ＩＭＰで一連番号を処理することによって，最初の２つの考慮事項(1)，(2)を処理することができる。又，再組立てをホストがやるならば，ホストは，一連番号を割当て，処理し，重複パケットをチェックすることができる。多くのアプリケーションにおいて，目的地に運ぶメッセージの順番は，重要でない。しかし，優先メッセージの場合，早い順番で配達しなければならないので，メッセージの順番を乱してしまう。

通信回線上の雑音が原因でメッセージにエラーが入りこんだ場合には，エラーを検出し，伝送経路上にあるＩＭＰの各々の組の間で再送されることによって簡単に処理できる。もし，回線速度か，回線の長さのどちらかが，実質的に増加すれば，ＩＭＰ中に余分の記憶域を必要とする。メッセージに小量の余分のオーバエッド（ARPANETでは，50キロビット／秒の伝送回線を使用して，パケット当り，20 － 30 パリティ・ビット）を入れておけば，エラー検出の障害は１世紀に１度位しか起らない。標準サイクリック・エラー検出コードが，ここでは適用されている。

信頼度の高いシステム設計は，伝送される各々のメッセージが，意図する目的地に正確に運ばれることを保証している。

ＩＭＰが故障し，有効な制御（in-transit）メッセージをこわすような事は，多分，ホストでの同様な故障よりもかなり少ないし，ＩＭＰのメモリ・フェイラーによるエラーのように，実際には重要でないことが証明されている。簡単な端と端（end to end）での再送機構は，もし，実際に必要になったならば，これらの情況を保護することができる。しかし，ＩＭＰは，内部的に格納されたパケットをこわすことなしに，ネットワークからエラーを取り去ることができるように設計されている。

C．フロー制御

パケットが，自由に入ったり出たりできるネットワークは，輻輳（congestion）したり，論理的にデッドロックしたりする。それは，トラフィックの移動を停止する原因となる。フロー制御手法は，これらの情況が起こらない

ネットワークが，初期に受入れなければならないトラフィック量はあらかじめ正確にはわからないので，普通，妥当なトラフィック量を受入れるに十分な余分の容量を見込んでネットワーク建設が行なわれる。そのとき，新しいＩＭＰがシステムに増設されたとき，トラフィック・レベルがネットワークの全容量のかなりの部分を占有するまでは，ネットワーク全体の容量は組織的に減少する。この時点で，ネットの容量が望ましい負荷率を保持できるように拡大するため改良される。ネットワーク設計の初期段階では，"２点間結合（two-connected）"の信頼度の制約が，本質的に最高スループット率の最小値を決定する。すべてのIMP対間に２つの通信経路が必要であり，50キロビット／秒の回線がネットワークを通して使用できるとき，この制約が，平均スループット率をＩＭＰ単り，10～15 キロビット／秒の範囲におさえてしまう。言い換えると，もし，このレベルのスループット率を要求するならば，そのときは，"２点間結合度（two-connectivity）"の信頼度の実現は，付加コストなしで得ることができる。

Ａ．信頼性計算

ネットワークの信頼性の特性は，簡単に言うと，すべての運用可能なＩＭＰ対間の通信を持続するためのネットワークの能力ということができる。設計目標によって，ノード間には，２つの独立な経路があるので，２個のＩＭＰおよび回線が同時に故障しないかぎり，運用可能なＩＭＰの任意のペアーが通信できなくなることはない。

この規準は，ＩＭＰと回線の特性に無関係で起るかもしれない切断の程度を考慮しなくてもよく，そして，それ故に，ネットワーク中のリソースの実際の可用性に影響を及ぼさない。もっと意味のある測度は，ＩＭＰと回線の故障のために通信できないＩＭＰ対の平均率である。この計算には，ＩＭＰと回線の故障率が必要であり，正しい推定を行なうために十分な運用データが収集されるまでは，この計算を行なうことはできない。

ネットワークの信頼度を計算するため，切断集合*（cutsets）として知

＊それを取除くことによってフローの流入点と流出点とが切離されてしまうようなリンクの集合

られている基本的なネットワークの構造について考察しなければならない。切断集合は，すくなくとも２つの運用可能なＩＭＰ間のすべての通信経路を切離し，ネットワークから取去った回線網とＩＭＰの集合である。信頼度を計算するために，すべての切断集合は，正確に数えられるか，概算できるかのどちらかでなければならないというケースがしばしばある。例として，23個のＩＭＰと28個の回線のＡＲＰＡネットワークの設計において，のこりのネットワークが，すくなくともno-intactな通信経路を持つＩＭＰの１つの運用可能なペアーをもつように回路網だけを削徐する組合わせは，２千万通り以上ある。表５－２は，ネットワークが含んでいる回線網の数を関数にして，23個のＩＭＰネットワークでの切断集合の個数を示している。不通（non-communicating）のIMP対の平均率を計算するために解析とシミュレーションの両方が使われる。ネットワーク中に23個のIMPがある場合の斯析結果が，図5-23に示されている。Aの印のついた曲線は，IMPが故障しないと仮定したときの結果であり，一方，Bの印のついた曲線は，回線網が故障しないケースである。Cの印のついた曲線は，ＩＭＰと回線網の両方が等確率で故障を起すと仮定した。実際の運用において，ＩＭＰと回線網の両方の平均故障率は，およそ0.02である。この値では，回線網の故障の影響は，ＩＭＰの故障の影響よりもずっとすくない。もし，ｎ個のＩＭＰをもつネットワークで，１つのＩＭＰが故障すれば，すくなくとも，他のｎ－１個のＩＭＰは，それと通信することができない。このように，ネットワーク設計がよくても，ＩＭＰの故障による直接の影響を改善することはできない。

このことは，ARPANETにおいて通信の信頼度に影響をあたえる主要な要因である。さらに，又，回線網の故障によるスループット量の損失を考慮したもっとも複雑な信頼性計算が行なわれ，これらの損失は無視できることが示されている。最後に，回線長の違いによる故障率の差異は，解析上でははほとんど重要でなく，普通は無視することができる。

表5－2　23 Node 28 Link ARPA

Number of Circuits Failed	Number of Combinations to be Examined		Number of Cutsets
28	1	same	1
27	28		28
26	378		378
25	3276		3276
24	20475		20475
23	98280		98280
22	376740		376740
21	1184040		1184040
20	3108105		3108105
19	6906900		6906900
18	13123110		13123110
17	21474180		21474180
16	30421755		30421755
15	37442160		37442160
14	40116600		40116600
13	37442160		37442160
12	30421755		30421755
11	21474180		21474180
10	13123110		13123110
9	6906900		6906900
8	3108108		3108108
7	1184040		1184040
6	376740		349618
5	98280		≈70547
4	20475		≈9852
3	3276		827
2	378		30
1	28		0

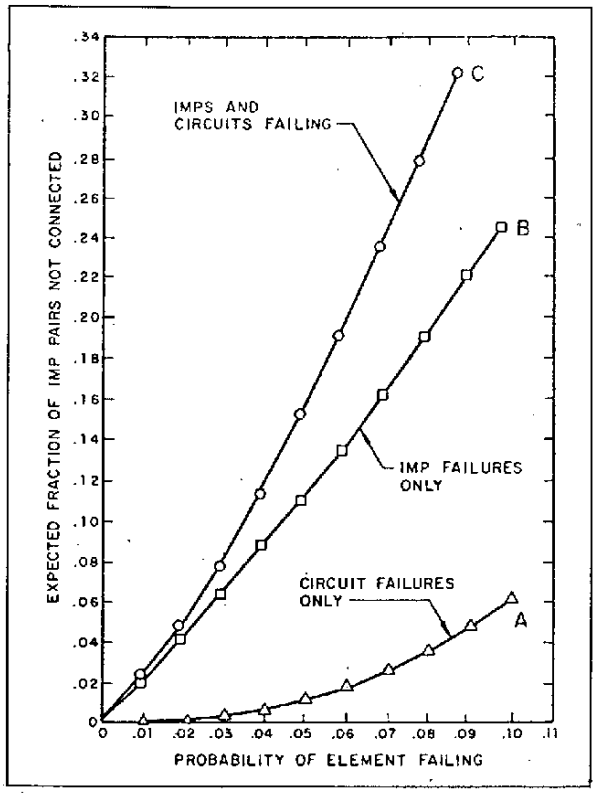


図5－23　Network availability vs. IMP and circuit reliability

B．トポロジカルな最適化

コンピュータによる最適化の過程で，トポロジーの信頼度は，もし，ネットワークがすくなくとも2本で連絡されていれば通信できると仮定する。最適化の目標は，全体の費用を制限しておいて，コストとスループット比を減小することである。この技法は，低コスト設計を行なうために回線交換ヒューリスティック，経路選択とフロー解析アルゴリズムを利用する精巧なネットワーク最適化プログラムを使用している。加えて，

(1)　平均遅延時間を0.2秒に保つスループト率の計算

と，

(2)　ＩＭＰ中のすべてのバッファが一杯のためによるパケット棄却率の推定

を行なうための2つの遅延時間モデル（後述）は，初期には使われた。この

モデルによる実験が行なわれて，パケット棄却率は，無視してもよいことがわかり，計算を中断した。その後，遅延時間の計算式（後の項の式(7)）は，計算時間がかかりすぎるので，ヒューリステック計算に置換えた。さらに，切断集合が飽和しなければ，遅延時間がかなり小さいことがわかったので，遅延時間の計算式は，その後使われなくなった。"閾値的"な振舞いは次の項で，さらに検討される。

最適化の基本的な方法は，全体のネットワークの制約条件を満足する"ネットワークの初期モデル"（starting network）を最初に，マニュアルまたはコンピュータのどちらかで発生することによって作動する。しかし，そのネットワークの初期モデルは，一般に低コスト・ネットワークでない。コンピュータは，制約条件を満足するより低コストのネットワークがみつかるか，あるいは，プロセスが終了するまで，簡単な手段でネットワークの初期モデルを繰返し変更する。その処理も改良すべき点がみつからなくなるまで繰返される。　別の初期値ネットワークを使用すると，別の解が得られる。しかし，気のきいたヒューリスティックを結合し，注意深く選択された初期値ネットワークの組合せと何回かのマン－マシン・インタラクションを使うことによって，"最適な"最終のネットワークがみつかる。しかし，異なったトポロジカル構造で，同様なコストとパフォーマンスをもつような多くのバラエティに富んだネットワークがあることが，実験によって示された。

この設計を効果的に行なう鍵は，何回もネットワークの修正を行なうヒューリスティック・プロセデュアである。ARPANETで使用された方法は，同時に1個又は2個の回線網を削徐したり，追加したりする機能を含んでおり，あらゆる機会に，多くの方法が，可能性のある追加又は削徐行なって，最適な回線網をみつけるため利用された。例えば，簡単なプロセデュアは，メガビット単りのコストの減少がみつかっている間は，もっともコストのかかる回線網1個を削徐し，トラフィックに対して再経路選択を行ない，コストの再推定を行なう。この時点で，削徐はくり返し行なわれ，その処理を再び実行する。ちょっと複雑なプロセデュアは，利用率を増加するため回線網を

削徐する。一方，もっと複雑な方法は，削徐を行なう前に，ある回線網の削徐の影響を推定することを試みている。そのとき，もっとも改良の可能性のある回線網が，削徐するために検討される。以下，同様にこの処理をつづる。

回線交換に関して多くの有効なヒューリスティックがある。これらの方法を使っての詳しい実験を行なった後で，特殊なヒューリスティックの選択はクリティカルでないことがわかった。変わりに，ポテンシャルな交換を研究するときに使う速度と効率は，最終設計に影響を与える制約要素であることがわかった。結局，ネットワークの規模が拡大するにつれて，回線交換の最適化を実行するのにより多くのコストが必要になる。したがって，ネットワーク設計をregions（領域）に分解することが必要となり，付加的なヒューリスティックは，効果的な分解を決定するのに必要になる。現在，これらの方法は，2～300個のＩＭＰを含む相対的に効率のいいネットワークを設計するのに使われ，一方，強力な新しいプロセデュアは，大規模のネットワークに使われている。

トポロジカル設計は，与えられたネットワークのスループット容量を評価するための経路選択アルゴリズムを必要とする。その特性は，例えば，ARPANET中のインプリメントできる経路選択アルゴリズムのどれかに反映しなければならない。経路選択問題は，"多種フロー問題（multicommodity flow problem）"として公式化され，20－30 個のＩＭＰをもつネットワークは，線形計画法によって解かれているが，より高速な解法が，経路選択アルゴリズムを設計プロシデュアに組込むとき，必要である。ＡＲＰＡネットワークのトポロジー設計プロセデュアは，数千のネットワークを繰返し解析する。速度に対する要求を満足させるため，ＩＭＰの最小番号をもつ，もっとも最近に利用した経路を選択するアルゴリズムが，初期に使用されていた。後に，このアルゴリズムは，１つ以上の回線網がフル利用率の数パーセントになるまでは，できるかぎりたくさんのトラフィックをそのような経路に沿って送出するアルゴリズムと置き換られた。そのとき，これらのより高利用率の回線網は，もはや，それ以上のフローを流し得ない状態になってい

る。変わりに，余分の容量をもつ新しい経路と出来るだけ近くにあるノードが見付けられる。そのプロセデュアでは，ある切断集合が，ほぼ100％の利用率である回線だけを含むまで，この操作をつづけられ，この時点で，追加フローを送出することができなくなる。設計目標に対して，このアルゴリズムは，もっと複雑な多種フロー問題のアプローチに対する高度に十分な置換法である。このアルゴリズムを使用して，ＡＲＰＡネットワークのスループット容量は，トラフィックの分布に非常に敏感で，主として，ネットワーク中の全トラフィック・フローだけに依存することが示された。

5.2.3　ネットワーク・パフォーマンスのアナリテック・モデル

システム・パフォーマンスのアナリテック・モデルを決定するための作業は，次の２つのフェーズに分けられる。

(1)　メッセージが，ネットワークを通過するときのメッセージの平均遅延時間の予測。

(2)　実現可能な最小遅延時間に対する通信路容量の最適割当てを計算するための，これらの待ち行列モデルの使用。

A．単一サービス・モデル

待ち行列理論[18,22]は，パケットの遅延時間を研究するための解析的な道具である。この理論の大部分は，単一通信路（単一サービス）で，メッセージが伝送（サービス）要求をするようなシステムについて考察している。これらのシステムは，A（τ）とB（t）で記述される。ここで，

A（τ）：要求の内部到着間隔の分布

B（t）：サービス時間の分布

である。平均サービス要求が，通信路（チャンネル）の容量以下であるとき，システムは，定常状態（stable）であるといわれる。

A（τ）が指数分布（すなわち，ポアソン到着）で，メッセージは先着順サービスで伝送される場合，定常状態にあるシステムの平均遅延時間Tは，

$$T = \frac{\lambda \overline{t^2}}{2(1-\rho)} + \overline{t} \qquad (1)$$

である。ここで，λはメッセージの平均到着率，$\bar{t}$と$\bar{t^2}$は，各々，B(t)の1次と2次のモーメントであり，そして$\rho = \lambda \bar{t} < 1$である。又，もし，サービス時間が，指数分布ならば

$$T = \frac{\bar{t}}{1 - \rho} \tag{2}$$

でとなる。

A(τ)とB(t)が任意の分布の場合，状態が複雑になり，単に，不十分な結果が得られるにすぎない。例えばTに対する式は利用できなくなる。しかし，$\rho \to 1$とすれば，次の上限条件式は，正確な近似となる：

$$T \leq \frac{\lambda(\sigma_a^2 + \sigma_b^2)}{2(1-\rho)} + \bar{t} \tag{3}$$

ここで，σ_a^2とσ_b^2は，各々の内部到着時間とサービス時間分布の分散である。

B．待ち行列ネットワーク

ネットワーク・エンバイロメンでの多重通信路は，単一サービス系よりもっと解くのに複雑である待ち行列問題となる。例えば，メッセージの発信地と目的地の選択の可変性は，遅れを引き起すネットワーク現象である。原理的な解析の困難さは，ネットワークを通過するフローが，相互に依存するという事実による。これらのストカスティックなネットワーク問題を解く基本的なアプローチは，もとのネットワーク構造とトラフィック・フローを反映する解析可能な単一サービス問題に分解する。

待ち行列ネットワークの初期の研究は，そのような分解が可能であることを示した。しかし，これらの結果は，トラフィック・フローの相互作用があるメッセージ交換コンピュータ・ネットワークに適用することはできない。各種の通信網でも，パケットが，ノードからノードへ通過するとき，パケット長を独立な確率変数とみなせば，この相互作用を削徐できることが，示されている。この"独立"という仮定は，物理的には実在しないが，それは，結果として，予想遅延時間の精度に影響を与えないと思われる。数学的に取り

扱いやすいモデルとなる。ネットワークの大きさと結合度が増すと，この仮定はもっと現実的になってくる。この仮定を使って，ネットワークを適切に分解すれば，各通信路毎の解析が可能である

パケットの遅延時間は，パケットがその発信地から目的地に到着するまでにネットワーク中で経過した時間と定義する。平均パケット遅延時間をTとする。Z_{jk}は，発信地がIMP_jで目的地はIMP_kであるそれらのパケットの平均遅延時間とする。発信地からのパケットの到着が，平均到着率γ_{jk}パケット／秒のポアソン分布であり，パケット長が平均パケット長$\frac{1}{\mu}$ビット／秒の指数分布に従うと仮定する。これらの定義を使って，γがγ_{jk}の総和であるとすれば，そのとき，

$$T = \sum_{j,k} \frac{\gamma_{jk}}{\gamma} Z_{jk} \tag{4}$$

式(4)を，単一通信路遅延時間の項で書き直しをする。

まず，第i番目の通信路に対して，次の量を定義する。

C_i：第i番目の通信路容量（ビット／秒）

λ_i：第i番目の通信路が運ぶ平均パケット・トラフィック率（パケット／秒）

T_i：パケットが第i番目の通信路をあくのをまち，そしてその通信路を使用して経過した平均時間

経路選択アルゴリズムによって選択された経路の回線利用率λ_i/γ_{jk}を使うと，

$$T = \sum_{i} \frac{\lambda_i}{\gamma} T_i \tag{5}$$

となる。トラフィックの到着がポアソン分布で，サービス時間が指数分布に従っているとき，T_iは式(2)で与えられる。平均パケット長が$1/\mu$ビットで，$\bar{t} = 1/\mu C_i$であるとき，

$$T_i = \frac{1}{\mu C_i - \lambda_i} \tag{6}$$

このように，解析問題を1つの単一通信路問題に分解するとうまく解ける。

強力な分解は，パケット長がポアソン分布でなくてもよく，トラフィックのマルコフ性が保持されると仮定して，T_i の近似計算をするために，式(2)よりむしろ，式(1)が利用される。さらにその上，コンピュータ・ネットワークの場合，パケットの平均遅延時間に対する次の式を得るために，伝播時間とオーバヘッド・トラフィックを勘定に入れてある。

$$T = K + \sum_i \frac{\lambda_i}{\gamma} \left[\frac{1}{\mu' C_i} + \frac{\lambda_i / \mu C_i}{\mu C_i - \lambda_i} + P_i + K \right] \tag{7}$$

ここで $1/\mu'$ はホスト・パケットの平均パケット長を表わし，そして，$1/\mu$ はネットワーク中の（ＡＣＫ信号，ヘッダー，RFNM信号，パリティ・チェック，その他を含む）すべてのパケットの平均パケット長を表わす。

式 $1/\mu' C_i + \{ (\lambda_i/\mu C_i)/(\mu C_i - \lambda_i) \} + P_i$ は，第 i 番目の通信路でのパケットの平均遅延時間を表わしている。項 $(\lambda_i/\mu C_i)/(\mu C_i - \lambda_i)$ は，第 i 番目の通信路が利用可能になるまで，ＩＭＰでパケットが待っていた平均時間である。パケットは，ＡＣＫ信号と他のオーバヘッド・トラフィックと競合しなければならないので，オーバヘッド・パケットの平均長 $1/\mu$ が，式の中に現われる。項 $1/\mu' C_i$ は，平均長 $1/\mu'$ のパケットを伝送するのに必要な時間である。最後に，Kはノードの処理時間で，一定と仮定し，そして，ARPA IMP の場合は，約0.35 ミリ秒であり，P_i は，第 i 番目の通信路での伝播時間（およそ，3000 マイルの通信路では20ミリ秒）である。

C_i と P_i の対の集合が相対的にみて同次であると仮定すれば，遅延時間の式中の各々の項は，１つの通信路（通信路 i_0）中のフロー λ_i/μ が，容量 C_{i_0} に近ずくまでは総和を左右しない。その時点で，項 T_{i_0}，ここではＴが急速に増加する。遅延時間の式は，そのとき，１つ（又はそれ以上）の項に左右され，閾値的な振舞いを呈する。この閾値現象が起る前に，Ｔは相対的に一定値を持続する。

この閾値現象と同様に，遅延時間モデルの精度も，19ノード・ネットワークや図５－２(C)から右側にある５個のＩＭＰを削除した10ノードＡＲＰＡネッ

トでも論証された。最後の項で述べる経路選択プロセデュアと，すべてのノード対間が等トラフィックであることを使って，通信路フロー λ_i の値は，10ノード・ネットの場合がみつかっており，図5－24に示されている遅延時間曲線が求まっている。

曲線Aは，1000ビットの固定長パケット*で求めたものである。一方，曲線Bは，パケット長が平均500ビットである指数分布に従う可変長パケットに対して求めたものである。2つのケース，AとBにおいて，すべてのオーバヘッド要素は無視されている。全スループット率が，400キロビット／秒以上になるまでは，遅延時間が小さいということを明記しておく。そのとき，遅延時間は，急速に増加する。曲線CとDは，各々，パケット当り136ビットのオーバヘッド，RFNM信号当り136ビットのオーバヘッド，そして，ACK信号当り152ビットのオーバヘッドが含まれるときの同じ状態を表わしている。IMP当りの全フループット率は，ケースCでは250キロビット／秒に，ケースDでは約200キロビット／秒に減少する。

同じ図に，実際の経路選択と測定機構を使用して実行したシミュレーションの結果を⊗印を付けて図示した。シミュレーションは，すべてのネットワーク・オーバヘッドを無視し，すべてのパケットは，1000ビットの固定長であると仮定した。

各IMPが，同じ量だけのトラフィックを入力すると仮定すれば，実際的な経路選択とフロー制御プロセデュアを開発することは困難である。遅延時間曲線Aとシミュレーションによって得られた点を比べるため，その曲線は，結果として求まったほんのすこし曲がった分布に対して，実際に再計算された。遅延時間は，（動的経路選択方式を使用した）シミュレーションから見積られ，（静的経路選択方式と遅延時間計算式を使用した）計算値とより一致する。特に，それらは，およそ400キロビット／秒の範囲内にある遅延時間曲線の垂直立上り，有限の遅延時間を予測する公式，そして，それ以上のトラフィックの流入を許さないシミュレーションを正確に決定する。

＊Aのケースでは，式(1)を一定パケット長（すなわち，分散がゼロ）に適用する。

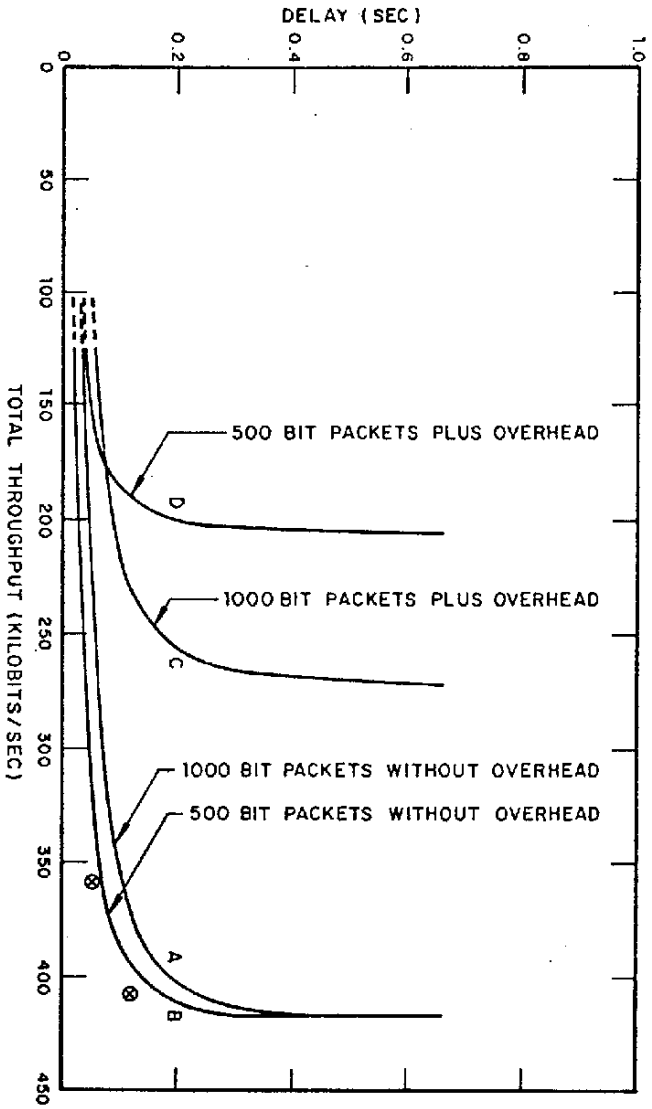


図5－24　Delay vs. throughput

ARPANET に対する解析やシミュレーションの研究から平均待ち行列遅延時間は，ほんのすこしだけ（ほとんど，無負荷ネットのそれ）あることが観測された。ネットワーク中のトラフィックが切断集合の容量に近ずくまでは，遅延時間は設計時に制約された0.2秒の範囲内である。それを越えると，遅延時間は急速に増加する。この様に，トラフィックが十分小さく，かつ，経路選択が，容量を越えた経路を通るトラフィックを別の経路に振分けることによって，切断集合の永久飽和を完全にさけることができれば，待ち行列遅延時間は，重要でない。

C．通信路容量の最適化

設計問題でもっとも困難なものの１つは，有限のオプション集合から最適容量を選択することである。この問題に対するたくさんのヒューリステックなアプローチがあるが，解析的に結果が求まるのは，比較的まれである（中央制御方式のネットワークの特殊なケースの場合，最適な結果をもたらすアルゴリズムが利用できる。）例えば，200ＩＭＰネットワークの場合には，不連続な容量の経済的割当てをみつけることが可能であるが，そのような容量割当て，メッセージ遅延時間，そしてコストの関係については，ほとんど知られていない。

最適容量割当ての理論的な特性をみつけるために，容量は不連続な大きさの値しかとらないという制約を無視する。ネットワーク・トポロジーと平均トラフィック・フローが既知で，固定で，そして，トラフィック・フローに対する容量の最適割当てがわかっていると仮定した場合の問題を提出している。又，トラフィックは，マルコフ性（ポアソン到着でパケット長が指数分布）で，独立に取り出して解析できると仮定すれば，分解法が適用できる。各通信路に対して，システムの全コストを一定にし，メッセージの平均遅延時間Ｔを最小にする容量C_iがみつかっている。λ_i/μ は，第 i 番目の通信路の平均ビット率であるので，任意の最適割当て問題に対する解は，各通信路に対して最小容量以上を割当ててしまうに相違ない。式(6)と(7)の両方は，T_i がこの容量以下の任意の大きさになることを示しているので，このことは明らかである。$C_i > \lambda_i/\mu$ であるかぎりは，どの位の余分の容量が割当てられるかは，厳密にはクリティカルでない。他の重要なパラメータや考え方が連続的な容量の最適化問題の研究で判明した。例えば，最小容量 λ_i/μ が，各通信路に割当てたあとに残った超過分のコストの大きさDeは，非常に重要である。$De \to 0$にしたとき，平均遅延時間は法外に大きくなるにちがいない。この範囲内では，ρ と $\bar{n}$ がクリティカル・パラメータである。ただし，ρ は，ビットがネットワークに入ってくる到着率 γ/μ とネットがビットを処理できる処理率Ｃの比，すなわち，$\rho = \gamma/\mu C$ と定義する。また，$\bar{n} = \lambda/\gamma$ と

定義する。λは，パケットがネット内を流れるときの到着率の総和，すなわち，$\lambda = Z\lambda_i$と定義する。ρは，ネットワーク"負荷"で，単位は無次元である。それ故に，$\bar{n}$は，パケットに対する平均経路長を表わすのによく使われる。

負荷ρが$1/\bar{n}$に近ずくにつれて，遅延時間Tは，非常に急激に増加する。そして，$\rho = 1/\bar{n}$の点で，ネットワークがサポートできる最大負荷を示す。容量が最適に割当てられたとすると，すべての通信路は，この時点で，同時に飽和する。この式で，$\bar{n}$は，トポロジーと経路選択プロセデュアを定める設計パラメータで，一方，ρは，入力率とネットワークの全容量を定めるパラメータである。ARPANETの研究で，そのネットワーク中で使用された高速電話データ用通信路に対する実際のトラフィックのより厳密な表現は，$D = \sum_i d_i C_i$と置くことによって与えられる。ただし，$0 \leq \alpha \leq 1$とする*。このアプローチは，数値解析によって非線形方程式の解を求める必要がある。方程式を解くと，パケット遅延時間Tは，$0.3 \leq \alpha \leq 1$の範囲にあるαによって変わることがわかる。これは，$\alpha = 1$で，前に検討した厳密な式の解が，もっと複雑な非線形問題に対するかなりよい近似値であることを示している。これらの連続的な容量の研究は，一般のネットワークの研究（例えば，衛星通信）に適用され，そして，現在も研究中である。

実際上，通信路容量の選択は，1つの小さい有限集合から選ばなければならない。いくつかの理論的な研究は，連続的な容量によって，不連続のコスト－容量関数を近似することによって，このケースに適用された。不連続な容量とネットワーク・トラフィックの時系列特性のため，通信路容量を通信路内の前もってわかるフローに適合できるようにするのは，一般に不可能である。もし，これが可能ならば，すべての通信路は，外部的に適用されたロードの同じ値で飽和する。前もってフローがわかっていない

＊もちろん，トラフィックは，利用可能な通信路の不連続な特性を反映する。指数αの使用は，不連続なコスト関数に連続的な適合を与える。ARPANETの場合は，$\alpha \cong 0.8$。

ので，容量は，平均，又は，ピーク時のトラフィック・フローの正当な推定値にもどずいて割当てられる。それは，トラフィックを利用可能な容量に合うよう振分ける経路選択プロセデュアの応答性である。しばしば，２つのＩＭＰサイトで大量の通信を行ない，１つ以上のクリティカルなネットワーク切断集合を飽和させてしまう。そのようなケースでは，経路選択は，追加フローをこれらの切断集合を横切らないように運ぶ。それ故に，ネットワークは，前述した閾値的な振舞いを導く通信路の１つ，又は，小さな集合で“早期”飽和に遭遇する。

5.3 パフォーマンスの測定（ARPANETの場合）

コンピュータ・ネットワークの測定の目的は，ネットワークの振舞いを把握できると共に，アナリテック・モデル，あるいは，シミュレーション・モデルなどのモデルの正当性をチェックすることも可能である。実際のネットワーク・エンバイロメントにおいて，適切なトラフィックによるテストを実行することによって，モデルの検定や改良の方法を提供する。

以下，ARPAコンピュータ・ネットワークのＩＭＰ中に組込まれている測定の為の道具（統計ルーチン）と，その測定の適用例のいくつかを紹介する。

5.3.1 測定器具

適当なサイトを選んで，以下に示すようないくつかのデータ収集ルーチンを動かすことによって，ＩＭＰはネットワークの使用状態と特性に関する統計量を集めることができる。

- 累積統計（回数とヒストグラム）
- スナップ・ショット統計（待ち行列の長さと経路選択表等）
- トレース・データ（メッセージ・フローに対するイベントの時系列）
- 情況レポート（アクティビティとコンディション情報）

これらのうち，最初の３つのルーチンはNMC (Network Measurement Center)で広く利用され，一方，最初の３つのルーチンは，もっぱら，ネットワークの管理者であるＢＢＮで利用されている。

A．累積統計

累積統計ルーチンは，他の統計ルーチンに比べ，最も頻繁に利用される測定器具である。このルーチンは，各ノードにおけるアクティビティの集計レポートをつくり出すものである。この集計レポートには，ＨＯＳＴ→ＩＭＰ，ＩＭＰ→ＨＯＳＴ間のメッセージやモデム出力として送られるパケットなどの長さのヒストグラムばかりでなく，ＡＣＫ信号，ＲＦＮＭ信号，入力エラー，再送，送出語数などの個数，それらにかかわる諸々のデータに関するレ

ポートが含まれている。これらのデータは128秒毎にとられ，この間隔はカウンタのオーバーフローを利用して決められている。このデータは390バイトのメッセージとしてＮＭＣに送られ，そこで調べた後で，印刷して捨てられたり，後で利用するデータはファイルに記録される。

印刷されるときには，データには注釈が付けられたり，編集されたり，さらに計算処理が施されたりして図５－25に示すような印刷結果となる。以下に各フィールドの説明を示す。

パート1は，ＨＯＳＴ→ＩＭＰ，ＩＭＰ→ＨＯＳＴ間で転送されるメッセージ長に関する統計である。単一パケット・メッセージに対しては対数目盛，複数パケット・メッセージに対しては一様間隔目盛のヒストスグラムである。また，メッセージの最終パケットの平均語数も印刷されている。例では，人工的に220個のメッセージがジェネレートされて，ホストから受取られており，その内の208個は単一パケットで，12個が２パケットからなるメッセージであることを示している。

図５－25のパート２では，これらの220個のメッセージが，６つの目的地に等分に配分されていることがわかる。これらのサイトは，人工トラフィックがいくつかの蓄積交換ノードを通ってメッセージが送られる時の遅延時間の違いを示すために選択されたものである。

周遊時間（round trip time）は，発信地ＩＭＰでメッセージの最初のパケットが受取られてから発信地ＩＭＰにＲＦＮＭ信号が返ってくるまでの時間として測定されている。データ値としては，周遊時間の総和と周遊回数，それにＮＭＣで計算された平均値が示されている。

パート3には，本物のＨＯＳＴ，仮想ＨＯＳＴ両方に関するアクティビティが示されており，仮想ホストはＧＨＯＳＴ*として記されている。この表は，どのホストがHOST－to－IMPの振舞いに（パート１，２で示されている）

＊仮想ＨＯＳＴは，ＩＭＰ中のバックグランド・プログラムである。これらのプログラムは，統計ジェネレーション・ルーチン，デバック機能，とＩＭＰコンテール・テレタイプ・ハンドラの機能をもっている。

ACCUMULATED STATISTICS SOURCE = UCLA TIME = 29106 DATE = 04-17-71

1. MESSAGE SIZE STATISTICS

1.1 SINGLE PACKET MESSAGE

LENGTH (WORDS)	FROM HOSTS		TO HOSTS
0 - 1	0		0
2 - 3	23	******	0
4 - 7	29	********	0
8 - 15	53	***************	0
16 - 31	72	********************	0
32 - 63	31	********	0

1.2 ALL MESSAGES

LENGTH (PKTS)	FROM HOSTS		TO HOSTS
1	208	********************	0
2	12	*	0
3	0		0
4	0		1
5	0		0
6	0		0
7	0		0
8	0		0

1.3 AVE. # WORDS IN LAST PACKETS : FROM HOSTS = 17.7 TO HOSTS = 6.0

2. ROUND TRIP STATISTICS

DESTINATION	TOTAL R.T. TIME	# R.T.	AVE. R.T. TIME (MSEC)
UCLA	25	1	20.0
SRI	0	0	0.0
UCSB	1025	34	24.0
UTAH	2472	34	58.4
BBN	196	2	78.4
MIT	5211	35	110.3
RAND	0	0	0.0
SDC	0	0	0.0
HARV	0	0	0.0
LL	5704	33	130.4
STAN	0	0	0.0
ILL	5423	46	94.6
CASE	6565	36	148.6
CMU	0	0	0.0

3. MESSAGE TOTALS

	HOST 0	HOST 1	HOST 2	GHOST 0	GHOST 1	GHOST 2	GHOST 3
# INPUT MESSAGES FROM HOST	220	0	0	0	0	0	2
# CONTROL MESSAGES TO HOST	218	0	1	0	0	0	0

4. CHANNEL ACTIVITY

	# HELLO SENT	# IHY RECVD	# ACK SENT	# ACK RECVD	# PKTS RECVD	# MODEM ERRORS
CHANNEL 1 (TO SRI)	24	23	156	156	335	0
CHANNEL 2 (TO UCSB)	23	24	34	36	94	0
CHANNEL 3 (TO RAND)	23	23	65	39	127	0
CHANNEL 4 (TO)	0	0	0	0	0	0
CHANNEL 5 (TO)	0	0	0	0	0	0
TOTALS	70	70	255	231	556	0

	# RE-TRANS.	# RFNM SENT	# WORDS SENT	FREE LIST EMPTY	# BUFFERS REMOVED
CHANNEL 1 (TO SRI)	0	0	3159	0	0
CHANNEL 2 (TO UCSB)	0	0	604	0	0
CHANNEL 3 (TO RAND)	0	0	690	0	0
CHANNEL 4 (TO)	0	0	0	0	0
CHANNEL 5 (TO)	0	0	0	0	0
TOTALS	0	0	4453	0	0

5. PACKET SIZE STATISTICS (LEAVING THE IMP)

LENGTH (WORDS)	CHANNEL 1 (TO SRI)		CHANNEL 2 (TO UCSB)		CHANNEL 3 (TO RAND)	
0 - 1	14	******	4	*	5	**
2 - 3	9	****	6	**	2	
4 - 7	15	******	4	*	7	***
8 - 15	44	********************	8	***	7	***
16 - 31	44	********************	9	****	15	*****
32 - 63	31	**************	5	**	4	*
TOTALS PKTS	187		36		40	

図5－25 A Typical Accumulated Statiatic Print-Out.

寄与しているかを決めるのに役立つ。

個々のモデム・チャンネルEのアクティビティはパート4に示されている。HELLO信号とIHY信号（I Heard You）メッセージが定期的に送られ，そのとき回線がアクティビティである回数が示されている。（HELLOメッセージは経路選択表用の更新メッセージに含まれて計数されている。）全"#PKTS RECVD"数は，このHELLO信号メッセージ，ACK信号数，受信されたRFNM信号数，それに実際に受信されたメッセージ・パケット数を含んでいる。その他のデータは，送信したRFNM信号数，送信したデータ総語数，到着した時に入力バッファが満員であった回数（FREE LIST EMPTY），そして，そのケースで，バッファが獲得されるまでに蓄積交換用パケットを棄却した回数のそれぞれを示している。

図5－25の最終部分には，各モデム・チャネルでのパケット長のヒスグラムが対数目盛で示されている。これには伝送したメッセージの中味のみが記録されており，HELLO信号，IHY信号，ACK信号あるいはRFNM信号トラフィックは含んでいない。

これらの累積データの各々は128秒間隔以上のアクティビティの様相を示すのに用いられる。又，図5－25から，ホストからの全メッセージ数，最終目的地，平均周遊時間などを知ることができる。HOST－IMPのヒストグラムからはメッセージ長の分布に関する情報が得られ，モデム・チャンネル．のヒストグラムからはパケット長の分布と通路の利用率情報が得られる。この例によると，チャンネル統計は，UCLAのIMPにある経路選択表から予想した以上に，大量のパケットがSRI経由で伝送されている。これは，予想では，これらのパケットの約半分はRAND経由で送られることになっていた。この事実は，そのような振舞いの原因の究明に役立つ。このためには，スナップショットと呼ばれる別の測定器具類の中に含まれる経路選択表が必要である。

B．スナップショット統計

各スナップショット統計メッセージには，特定のＩＭＰに対して特定の時間間隔で測定された待ち行列長と経路選択表の情報が含まれている。これらの値は，0.82 秒間隔で記録され伝送される。この時間間隔は，状態変化の時系列の細部にわたり観測したいという要求と，その反面，この統計をあまりにもしばしば伝送すると測定系を乱してしまうという矛盾した問題のかね合いからきめられている。他の測定器具のすべてがそうであるように，スナップ・ショットを各々のＩＭＰに対して可能にすることもできるし，不可能にすることもできる。

C．トレース統計

トレース機能は，任意にネットワークを通りぬけていくメッセージを追跡したり，通過した経路や遭遇した遅延時間を知るための機能である。しかし，データの解釈は，再伝送の確率に依存すると同様に，ＩＭＰのメッセージ処理，すなわち，発信地から蓄積交換によって目的地までメッセージを運ぶ処理の関数に依存するので複雑である。

典型的な蓄積交換方式の場合には，

(1) 到着時刻

(2) メッセージが処理された時刻，すなわち，出力待ち行列に登録された時刻

(3) 伝送を開始した時刻

(4) ＡＣＫ信号を受信した時刻

の項目が記録として取られる。再伝送の場合には，この第４項の時刻が再伝送時刻になり，それは，データ・ブロックのタグ・ビットによって示される。目的地ＩＭＰでは，この第４項は，IMP－to－HOST伝送が完了した時刻である。すべての時刻は，100マイクロ秒の精度で計測され記録される。

トレース・データ・メッセージの別の情報は，パケットのすべての制御情報の入ったヘッダーと同様に，（モデム・チャンネルという条件で）出力チャンネル，ＨＯＳＴ番号，あるいは仮想ＨＯＳＴ番号を含む。ヘッダーは，

図５－26のような構成で，発信地，目的地，リンク番号，メッセージ番号，パケット番号，そして，優先／非優先状態等から成っている。

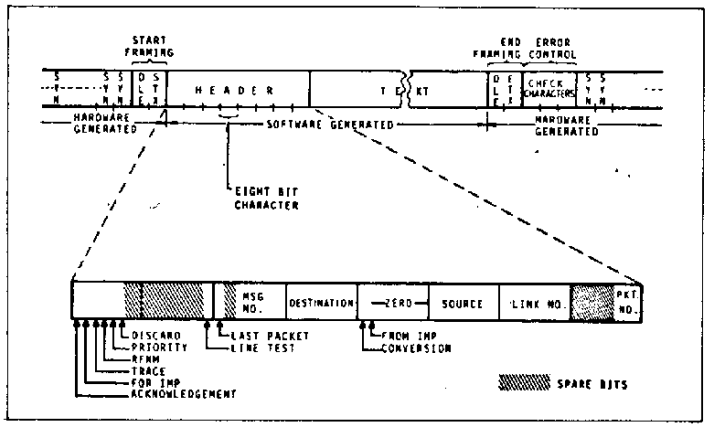


図5－26 Format of packet on phone line

D．人工トラフィックの発生

ネットワークの測定機能は，ＩＭＰとＮＭＣの両方にある人工トラフィック発生ルーチンによって遂行される。そのような人工トラフィックは，ネットワーク運用の初期段階におけるシステムの振落しテストに役に立つし，さらに，最近は，測定やモデリングに関する研究に必要な道具となっている。各ＩＭＰは，固定長メッセージからなる人工トラフィックを発生することができ，指定された時間間隔で周期的に１つのリンクへ伝送される。しかし，そのようなトラフィックは，ACK／RFNMオペレーションで組立てられてる現在の制御機構によるＩＭＰ上での輻輳現象をシミュレートするには十分でない。それ故に，もっと強力な人工トラフィック・ジェネレータが，ＮＭＣの研究の一部として開発された。そのジェネレータは，固定長又は不定長メッセージを内部伝送時間で63個までのリングを通してたくさんの目的地へ送る機能がある。図５－25の例で利用されたトラフィックは，メッセージ長ヒストグラムで示されるように不定長メッセージ発生の機能を利用して，この後者のジェネレータによって累積された。

5.3.2　測定結果

A．ネットワーク・アプリケーションに関する測定

この測定時点において，一般のホスト・ユーザによるアプリケーションはあまりなかったが，インタラクティブ処理，ファイル伝送，あるいは，グラフィック・ディスプレイ の利用のような機能を扱う特別な目的のプロトコル・サブセットを確立することによって，特殊なサイトのユーザは，このようなアプリケーションを利用した。

以下に，実際のネットワーク・アプリケーションで測定した例の詳細な測定結果を示す。

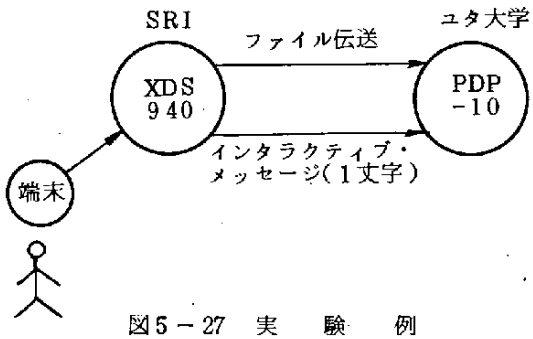


図5－27　実　験　例

ＳＲＩのＸＤＳ 940とユタ大学のＤＥＣ社PDP－10を次のようなインタラクティブ処理に使用する場合を考えてみる。

図5－27に示すように，まず，データ・ファイルをＳＲＩからユタ大学に伝送し，次に，インタラクティブに1文字をＸＤＳ 940からDPP－10へ伝送し，その応答をPDP－10ですぐ処理するインタラクティブ・オペレーションの例である。ファイルは，4608 ビットのブロックで伝送された。この大きさは，XDS 940の24 ビット語長とPDP－10の36ビット語長の適当な倍数であるし，又，PDP－10の最適なブロック・サイズ128語でもある。

プロトコルには，メッセージのタイプと長さを示すためいくつかの特別な文字が含まれる。その結果，1文字メッセージでも，データを伝送するときは，５ＩＭＰ語が必要である。これらの1文字づつとファイル伝送のプロト

コルの規約によって，表5－3に印刷されたサンプルが示すように，SRIからユタ大学への伝送のほとんどすべては，5語メッセージか，5パケット・メッセージのどちらかになる。各行は，1つの12.8秒間隔の統計メッセージを表わしている。time－of－dayは，データの値が連続して収集されて以来，開始時刻に12.8秒毎に加えた値を示してある。示されているイベントの系列としては，10語のファイル名メッセージ，67個の5パケット・メッセージ，4パケット・ファイル・フラグメント，"ene of file" メッセージ等のファイル伝送及びインタラクティブ・トラフィックの時系列から成っている。この特定のファイル伝送は，約39,300バイトで，約12.5キロビット／秒の平均帯域幅を使用すれば25.6秒間隔以上で起った。12.8秒間隔で観測された最大ファイル伝送帯域幅は，5パケット・メッセージの73個を伝送したときであった。各メッセージは，標準の576バイト・レコードである。総帯域幅使用は，

26.3 kb/sec	for data（実効データ率）[*1]
0.3 kb/sec	for marking,padding,and protocal format
3.9 kb/sec	for network header,checksum,etc.
30.5 kb/sec	total bandwidth used.

である。

ファイル伝送は，測定間隔である12.8秒の3倍以上かかっていることがわかる。メッセージの平均周遊時間は159ミリ秒である。最小周遊時間は，直列伝送と伝播遅延時間を考慮して，約138ミリ秒である。159マイナス138，すなわち，21ミリ秒の余分の遅延時間は，PDP－10のメッセージ受信処理における遅延時間である。実効周遊サイクルは，RFNM信号の発信地ホストでの処理と，発信地ホストから発信地IMPへのリーダー（leader）[*2]と

---

*1 $\frac{576\times73}{12.8}\times 8$

*2 メッセージの頭の32ビットで，目的地ホストとリンク番号が入っている。

最初のパケットの伝送に必要な時間として余分に12ミリ秒が必要である。それ故に，単一リンクでの最大データ率は，約150ミリ秒毎に576バイト・レコードが1個，すなわち，実際のデータの30.7kb／秒である。しかし，実際の観測によると，26.3kb／秒の最高実効データ率しかえられていない。測定された平均周遊時間と結果としての実効データ率を使用しての同様な計算は，約4から5ミリ秒が発言地ホストのＲＦＮＭ信号処理と次の伝送の準備を完了するまでに必要な時間である。

２つのＨＯＳＴ間でメッセージを伝送するのに必要な周遊時間は，メッセージ長Ｌの関数として図5－28にプロットされている。このスキャタ・プロットの各々の点は，１つのインタラクティブ・メッセージ，すなわち，１パケット・メッセージを表わしている。scatteringは，メッセージ長と周遊遅延時間とのバラツキの度合を表わしている。理論的にまとめた最小遅延

表5－3　A SUBSET OF THE ACCUMULATED STATISTICS MEASUREMENT OF THE SRI-UTAH TRAFFIC

time-of-day	avg. RT time	avg. # words	single packet messages * 0-1	2-3	4-7	8-15	16-31	32-63	multi-packet messages ** 2	3	4	5	6	7	8
21:59.990	153.7	39.0	0	0	0	1	0	0	0	0	0	29	0	0	0
22:00.203	118.1	32.4	0	1	9	file name				0	1	38	0	0	0
22:00.417	26.4	5.0	end of		32	0	0	0	0	0	0	0	0	0	0
22:00.630	0.0	0.0	file		0					0	0	0	0	0	0
22:00.843	82.3	5.0	0	0	11	interactive				0	0	0	0	0	0
22:01.057	23.9	5.0	0	0	6	traffic									
22:01.270	25.3	5.0	0	0	12	0	0	0	0						
22:01.483	27.6	5.0	0	0	19	0	0	0	0						
22:01.696	0.0	0.0	0	0	0	0	0	0	0	0	0	0	0	0	0
22:01.910	0.0	0.0	0	0	0	0	0	0	0	0	0	0	0	0	0
22:02.123	0.0	0.0	0	0	0	0	0	0	0	0	0	0	0	0	0
22:02.336	133.8	37.8	0	1	0	1	0	0	0	0	0	34	0	0	0
22:02.550	0.0	0.0	0	0	0	0	0	0	0	0	0	0	0	0	0
22:02.763	0.0	0.0	0	0	0	0	0	0	0	0	0	0	0	0	0
22:02.976	29.4	5.0	0	0	3	0	0	0	0	0	0	0	0	0	0
22:03.190	76.2	5.0	0	0	11	0	0	0	0	0	0	0	0	0	0
22:03.403	55.4	5.0	0	0	14	interactive				0	0	0	0	0	0
22:03.616	24.8	5.0	0	0	8	traffic				0	0	0	0	0	0
22:03.830	60.8	5.0	0	0	13					0	0	0	0	0	0
22:04.043	0.0	0.0	0	0	0	0	0	0	0	0	0	0	0	0	0

file transmission

* MEASURED IN TERMS OF IMP WORDS
** MEASURED IN TERMS OF THE NUMBER OF PACKETS PER MESSAGE

時間は，図５－28の点線で示され，観測された遅延時間は，この制約値以下でない。

この最小遅延時間の理論値には，以下のような要素から成る関数で

• 発信地ホスト — ＩＭＰ間のメッセージの直列伝送*遅延時間	＝	2.0＋0.32Ｌミリ秒
• パケットの伝播遅延時間とモデム遅延時間	＝	7.5 ミリ秒
• 目的地ＩＭＰでのパケット処理時間	＝	0.5 ミリ秒
• 目的地ＩＭＰ — ホスト間のメッセージの直列伝送遅延時間	＝	0.3＋0.16Ｌミリ秒
• ＲＦＮＭ信号の伝播遅延時間とモデム遅延時間	＝	7.5 ミリ秒
• 発信地ＩＭＰ — ホスト間のＲＦＮＭ信号の直列伝送遅延時間	＝	2.0 ミリ秒
• 発信地ホストでのＲＦＮＭ信号の処理時間	＝	0.2 ミリ秒
合計		20.0＋0.48Ｌミリ秒

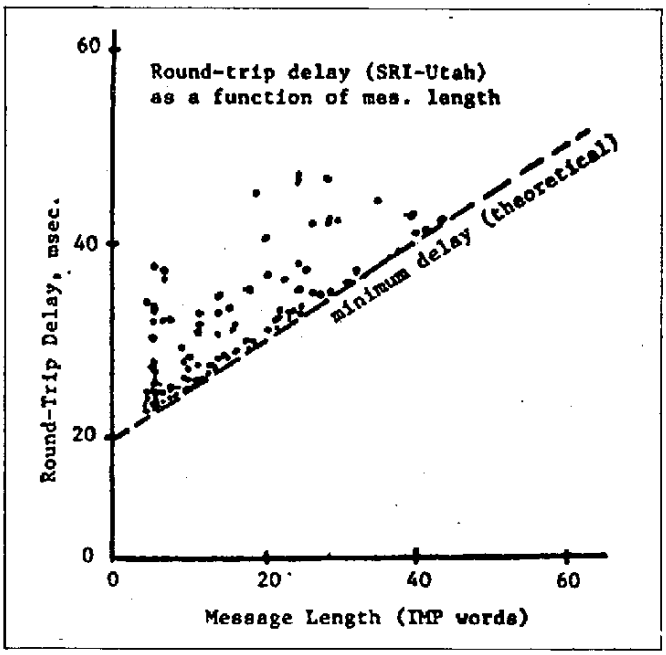


図5－28 Round－Trip Delay Measurements of the SRI－Utah Traffic.

＊ホストとＩＭＰ間のデータ転送は，ビット直列で全二重になっている。

あり，キューイングによる待ち時間は含まれていない。この全推定遅延時間は，図5－28に示された実験的に求めた最小遅延時間の値と非常によく一致している。図中の別の点は，待ち遅延時間，または，ＩＭＰからのメッセージを受取るときのホストの遅延時間によって，最小値を越えた遅延時間の値を取るメッセージを表わしている。

B．ネットワークのパフォーマンスと制約値を決定するための測定

これらの実験は，各種のトラフィック・レベルでのネットワークのパフォーマンスのさまざまな様相を評価したり，パラメータの感度や制約値を決定するのに役立った。次の例は，これらの実験の典型的なものであり，解析と測定技術の両方で考えられる迂回路があるような2つのノード間のスループット率の測定である。この例の目的は，2つの技術がネットワークの振舞いのアナリテック・モデルを改良したり検定したりするのに必要な見識をあたえるための測定法で，どの位補充されているかを示すことである。

初期のスループット試験は，ＵＣＬＡにあるＮＭＣからＵＣＳＢのＩＭＰ中にあるＤＩＳＣＡＲＤ*，すなわち，仮想ホストへ人工的なトラフィックを送り，伝送中に利用されたリンク数を関数として実効スループット率を測定することによって実施された。

ＩＭＰ間の通信回線は，50キロビット／秒の伝送帯域幅をもち，その値は1つの経路のスループットに対する上限の帯域である。しかし，各メッセージに関係するオーバヘッドは，図5－29に示すように，全スループット率をやや小さい値に制限する。同じ図に，トラフィック・フローが50キロビット／秒の値を越えた場合の迂回路のケースに対するトラフィック・フローの分枝による測定されたスループット率を示す。あらゆるケースにおいて，メッセージの伝送回数は，ＲＥＮＭ信号駆動（driven）率と同じで，すなわち，リンクの閉塞がとかれるやいなやメッセージは伝送される。

2つの異なったメッセージ長で上の試験が行なわれた。最初のケースに

＊ＩＭＰ中でパケットの棄却処理を行なうプログラム

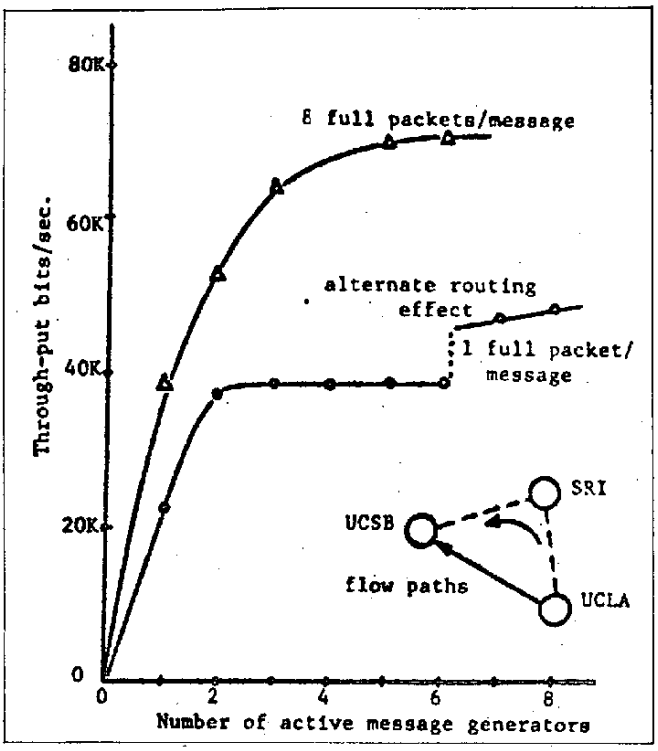


図5－29　Through-put Measurements Between UCLA and the Neighboring UCSB IMP

おいては，実効パケット・オーバヘッドが最小になる（パケット・サイズに関係する）ように，メッセージをフル・レングスの1パケット1個分の長さにした。そのようなメッセージに対する最大スループット率は，38.6 キロビット／秒と測定され，（意味のあるデータの伝送に20.0ミリ秒とパケットのオーバヘッドとＲＦＮＭ信号のＡＣＫ処理に5.7ミリ秒必要であることにもとずく）理論値 39.0キロビット／秒にまったく一致している。この最大スループット率は，単一リンクのケースでは，回送されるＲＦＮＭ信号を待っている間の静止期間があるため実現しない。この遅延時間は約 13.5 ミリ秒*で，この値は，" non-productive " な伝送期間に加えられ，結果として，単

＊　ＵＣＬＡからＵＣＳＢへは，320 ビット・メッセージでは 22.8ミリ秒の周遊時間が測定されており，そして，その長さのメッセージに対しては 9.3 ミリ秒の既知のサービス時間がかかることにもとずいている。

一リンクの予想スループット率25.5キロビット／秒になる。（測定値は，22.5キロビット／秒である。）1パケット・メッセージのケースに対するスループット率は，トラフィック・ジェネレータが6個から7個に変化したとき，ＳＲＩ経路の迂回路が取られるので，急に増加することが観測された。

フル・8パケット・メッセージのケースに対して，単一リンクのスループット率は，だいたい1パケット・ケースの飽和値，すなわち，38.5キロビット／秒に等しいことが観測された。この等しさは，ＡＣＫするためのＲＦＮＭ信号がより少なく，単一リンクの場合に起ったようなＲＦＮＭ信号の短い静止遅延時間がなくなるためである。迂回路は，2つのジェネレータがテストされるまでは観測されたかった。このことは，トラフィックが，直通のＵＣＬＡ－ＵＣＳＢ経路とＵＣＬＡ－ＳＲＩ－ＵＣＳＢの迂回路の間に一様に分割されたことを示している。

8パケット・メッセージのケースに対する飽和効果は，予想されたよりももっと低い帯域幅で起った。直通と迂回の経路の各々は，約43.5キロビット／秒（ＡＣＫするためのＲＦＮＭ信号がより少ないので1パケット・メッセージ・スループット率よりも高い）の実効率でメッセージを運ぶことができるからである。それ故に，上限値としては約70キロビット／秒と測定されたが，この現象の原因を明らかにするため，さらに調査された。この食違いに対しては，いくつかの理由が考えられ，以下に述べられているが，それらは，その様な伝送中の制約要素を表わしており，統計収集器具をこの問題に対して利用できる可能性を示している。

(1) 蓄積交換用バッファの不足

最高スループット率を得るためには，十分な量の蓄積交換用バッファがなければならない。なぜならば，経路選択表更新間隔（約520ミリ秒と考えられる）中，通信路をビジーに保つに必要十分な長さの待ち行列を創成しなければならないからである。それ故に，サービス時間として23.0ミリ秒必要なパケットの待ち行列は，全部で約23バッファをふさいでしまう。しかし，全部で25個の蓄積交換用バッファがあるので，これは，制御要素

であるとは思われない。又，スナップ・ショット測定は，典型的に18から20個の蓄積交換用バッファが使われていることを示している。このことは，この結論を実証していると思われる。

(2) 可能な帯域幅縮小効果

実効伝送帯域幅は，回線エラー，再組立て用バッファの不足と経路選択ループを含むたくさんの事項によって縮小することができる。しかし，測定は，これらの事項がスループットの減小の主要な原因となっていないことを示している。

(3) インタフェースとホスト・コンピュータ・システムの制約

ホストとホスト－ＩＭＰインタフェースの伝送容量は，91.5キロビット／秒の観測されたスループット率をもつローカル（ＵＣＬＡ）のDIS CARD，すなわち，仮想ホストへ向けて人工トラフィックを伝送することによって検定された。これは，ＵＣＬＡ-to-ＵＣＳＢの飽和値をかなり越えている。

(4) 詳細な振舞いを決定するためのトレース測定

累積統計とスナップ・ショットの両者により，ＵＣＬＡのＩＭＰ中に制約があることがわかったので，パケット処理の振舞いの詳細な調査が行なわれ，ＵＣＬＡのＩＭＰ上のすべてのパケットがトレースされた。（約100パケット／秒でジェネレートされ，各パケットは別々のトレース・メッセージになる。）このトレース・データは，上のパート(1)で計算の時に使われたＩＭＰのパラメータの値の中に２つのエラーがあることを指摘した。第１番目に，経路選択表更新が520ミリ秒のかわりに約640ミリ秒でおこることがわかった。このパラメータは，前は，520ミリ秒であると理解されていた。そして第２番目には，出力待ち行列上の蓄積交換用バッファの上限は，（25のかわりに）*に約20である。これらの新しいパラメータの値で

＊ＢＢＮの後のチェックは，実際の経路選択表更新間隔は，655ミリ秒であることを示している。蓄積交換用バッファの上限は，25（８進），すなわち，21（10進）である。

有限バッファ管理の効果について再考すれば，第１番目の通信路は経路選択表変更があったとき，約18個のパケットをサービスしなければならず，パケット当りのサービス時間を23.0ミリ秒として，通信路は，約420ミリ秒* の間ビジー状態であることがわかる。次の経路選択表更新は640ミリ秒後に起るので，第１番目の通信路は640ミリ秒の内220ミリ秒だけ休止状態（ inactive ）になる。実効スループット率の寄与は，

$$\text{スループット率} = \frac{420}{640} \times (43.5\text{キロビット}/\text{秒})$$

$$= 28.5\text{キロビット}/\text{秒}$$

全推定スループット率は72キロビット／秒であり，これは測定された72キロビット／秒に相当する。この事は，理論と測定の間の理由のある一致である。

このテスト・アクティビティは，各種の測定器具が予想されるデータ値とデータ実験値の間の差を一致させるためにどのように利用されたかを示すために詳細に述べた。

＊別の２つの蓄積交換用パケットは，SENT待ち行列に対して必要であることが仮定されている。

5.4　コンピュータ・ネットワークの改良（ARPANETの場合）

ＡＲＰＡネットワークは，コンピュータと高速の通信回線とを結合させた全国的な規模のシステムであり，過去３年間に40以上の独立したコンピュータ・システムを接続し，そのサイトは30個所以上にものぼっている。この間に行なった実験的なテスト・トラフィックやシミュレーションの両方の実験によって，ＩＭＰのソフトウエアには設計上の欠陥があることが解った。この欠陥は，非常に高いトラフィック・レベルになると明確になってくるものであった。従って，ネットワークの使用が高まることを予想して，ＩＭＰのプログラムを大巾に変更し，それらの欠陥を改良した。その他の多くのプログラムも，ＩＭＰのパフォーマンスの向上をはかるため変更された。以下に，改良したプログラムの構造について紹介し，特に，新しく開発したアルゴリズムとそのインプリメンテーションについてやや詳しく紹介する。更に，その結果得られたＩＭＰの現在のパフォーマンスについても紹介する。

改良後のＩＭＰシステムのパフォーマンスは，次のような向上が見られた。

- 蓄積交換用パケットに対するプログラムの処理時間は，20%だけ減少。
- 回線のスループット率は，516ＩＭＰの場合には４%，316ＩＭＰの場合には７%増大。
- 最大長のパケットに対する回線のオーバヘッドは，29%から16%に減少。

と改良されている。

5.4.1　新しいアルゴリズムの開発

バランスのとれた設計をした通信システムは，短いインタラクティブ・メッセージに対して迅速伝送，ファイル転送のような長いメッセージに対して高帯域伝送の機能をもっている。ＩＭＰプログラムは，これら２通りのトラフィック状態のもとで充分な性能が得られるように設計されている。ＡＲＰＡネットワークが稼動してからの最初の２年半の経験で，遅延時間を短かくという性能の目標は達成されていることがわかった。ネットワークの負荷が軽い時に

は，数個のＩＭＰを経由して短いメッセージを送り届けるのに約0.１秒しかかからず，ネットワークの負荷が重い時でも，その遅延時間はほとんどが0.５秒以下である。ネットワークは，低または中トラフィック・レベルで，長いメッセージに対して良好なスループット率を示した。しかし，負荷が重い時には，ネットワークのスループットは著しく低下し，高速伝送をもつという目標は完全には実現されなかった。

そこで，この初期のネットワーク設計の内，重負荷の場合，スループットが低下するという問題を単独に検討した。この問題は，目的地ＩＭＰが目的地ホストへメッセージを転送する転送率よりも早くメッセージが目的地ＩＭＰに到着してしまうことである。この状態を「再組立て輻輳」（ ｒｅａｓｓｅｍｂｌｙ ｃｏｎｇｅｓｔｉｏｎ）という。再組立て輻輳の結果として，目的地ＩＭＰは，そのホストに何もメッセージが送り込めないという状態を引き越す。これを「再組立閉鎖」（ ｒｅａｓｓｅｍｂｌｙ ｌｏｃｋｕｐ ）という。再組立て輻輳を回避するためのアルゴリズムとそれに関係のあるシーケンス制御のアルゴリズムを以下の小節で述べる。

さらに，ＩＭＰ間のトラフィックを処理するために必要なＩＭＰと回線の伝送速度は本質的に減らせることがわかった。この点における改良は，ＩＭＰが保持できる最大スループット率を増大した。この点における新しいアルゴリズムもまたあとの方で説明することにする。

Ａ．発信地 ― 目的地間のフロー制御

メッセージ・フローの効率をよくするために，ネットワーク中のどこかに，発信地ホストと目的地ホスト間である程度のバッファリングが必要である。その容量としては，ホスト間の通信路の伝送速度と通信路を経由する周遊時間を掛けた分だけあればよい。フロー制御の問題は，ネットワークのバッファリングが使用できないために，ネットワークを輻輳させ，再組立て閉鎖を引き越すようなメッセージがネットワークに入るのを回避することである。そのようなときの状態を図５－30に示す。

図５－30において，ＩＭＰ１はＩＭＰ３へ複数パケットのメッセージを送

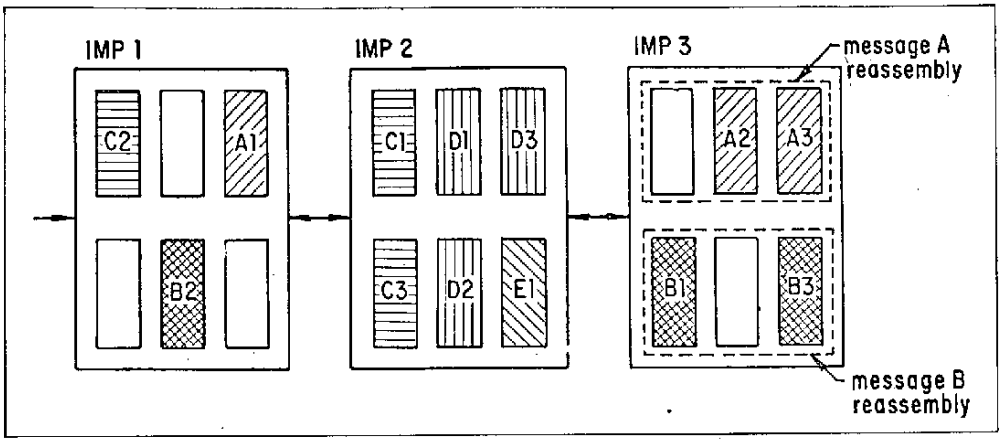


図5－30　Reassembly Lockup

ろうとしている。ＩＭＰ３の再組立て用バッファのすべてが，部分的な再組立てされたメッセージＡとＢで占有されているとき，閉鎖が発生する。ＩＭＰ３は，これらの部分的に再組立てされたメッセージの残りのパケットを待つため，残りの空地をすべて確保しているので，ＩＭＰ３はＩＭＰ２から送られてくるこれらの特定のパケットだけ受け入れることができる。これらの未済のパケットは，まだＩＭＰ１にあってＩＭＰ３に到着するにはＩＭＰ２を経由しなければならない。しかしながら，ＩＭＰ２は，ＩＭＰ３がまだ受信できないメッセージＣ，Ｄ，Ｅ（ＩＭＰ３へ送る）の蓄積交換用パケットでバッファが一杯になっているので，パケットＡ１，Ｂ２はＩＭＰ２を通ることは不可能である。以上のような情況から，ＩＭＰ３はメッセージＡとＢの完全な再組立てができなくなってしまう。

初期のネットワークの設計では，発信地 — 目的地のシーケンス制御とフロー制御をリンク機構に基づいていた。それぞれのリンクにおいては，１回に１個のメッセージしか許さず，リンク中でのメッセージの重複を検出するために一連番号を使用していた。

ホストが異常に多数のリンクにメッセージをまき散らせば，設計したフロー制御の機構が妨害されるということはあらかじめわかっていた。しかし，各ホストは問題が生じないようなレベルにリンクの数を押え，故意にフロー

制御の機構を混乱させるような行動はしないものと考えていた。しかし，シミュレーションや実験でネットワークの負荷を人工的に変えてみた結果，リンク数が適切であっても，ホスト間の通信が制御の機構を妨害する恐れのあることがわかった。さらに，各ホストがリンクを1つしか使用していない時でも，たくさんのホストが共通して1個所のホストと通信すれば，妨害が起り得ることが明らかになった。シミュレーションによれば，ある特定のホストに対して5個以上のリンクを同時に使用すると，再組立て閉鎖が起るということもわかった。複数パケット・メッセージで10個以上のリンクを使用すれば，再組立て閉鎖がほとんど即座に起る。

発信地ホストにバッファリングがあれば，伝送遅延時間が短くなるように最適化を行なうことができる。目的地ホストにバッファリングが備わっていれば，高速伝送を最適にすることができる。バランスのとれた通信システムを作りたいという考えにもとずき，再組立て輻輳に対する解決策として，発信地と目的地の両方にバッファ記憶装置を用いるという方法を開発した。この方法は発信地ＩＭＰから目的地ＩＭＰへの要求機構を利用している。*

ことに，複数パケット・メッセージに対しては，目的地ＩＭＰでメッセージを格納するための記憶域が割に付けられるまでは，そのネットワークに入ることを許されないようにした。

発信地ＩＭＰが複数パケット・メッセージの最初のパケットを受付けると，ただちに目的地ＩＭＰにこのパケット・メッセージに対する再組立て用記憶域を目的地で確保することを要求するため小さな制御メッセージを送る。発信地ＩＭＰがその要求に対する返答として割付け完了メッセージを受信するまでは，発信地ホストからあとのパケットを受付けるようなことはしない。目的地ＩＭＰはその要求を待ち行列に入れておき，十分な再組立て用記憶域

＊この機構は，Host-to-Host プロトコルのレベルで実験されているものとよく似ている。通信システムにおいてはどのようなレベルにおいても同じような問題が起るということを示している。

が空いた時，割付け完了メッセージを発信地ＩＭＰに送る。そこで，発信地ＩＭＰは目的地ＩＭＰにむけてメッセージを送り出す。

一連の長いメッセージに対しては，最初のメッセージを除いては要求機構を通らないようにして，メッセージの伝送効率を最大にしている。メッセージそれ自身が目的地に達し，目的地ＩＭＰがＲＦＮＭ信号を送り返そうとするとき，目的地ＩＭＰはつぎの複数パケット・メッセージ用の記憶場所が空くまで待機する。そのための場所があれば，ＲＦＮＭ信号に記憶域割付け完了のメッセージを付加して送り返す。もし，発信地ホストがＲＦＮＭ信号を受取ったとき，次のメッセージを発信すれば，記憶域の割付けは終っていて一度にメッセージを転送することができる。もし，発信地ホストがあまりにも長時間の遅れを生じたり，あるいは，データの伝送が終了したりしたときは，発信地ＩＭＰは，目的地ＩＭＰに対して割付け不用のメッセージを送り返す。このような機構を採用することによって，インター・メッセージ(inter message)遅延時間を最小にし，ホストはネットワークの全帯域幅を利用することができる。

短いメッセージについては，発信地ＩＭＰでコピーを保持している間に，ただちに目的地へメッセージを送ることによって遅延時間を最小にした。もし，目的地にそのための記憶域があれば，メッセージは受信され，ホストに渡され，ＲＦＮＭ信号が戻される。発信地ＩＭＰがＲＦＮＭ信号を受け取ると，保持しておいたメッセージのコピーを放棄する。そうでなければ，そのメッセージは無効となって，記憶域割付けの要求が待ち行列に入れられる。記憶域が利用できるようになった時，メッセージの再転送をするように発信地ＩＭＰに催促する。したがって，目的地で記憶域が使用可能になった時，セットアップによる遅延時間は生じない。

上述したような機構では，発信地ホストの実効伝送率は目的地ホストの受信率に等しいので，このＩＭＰネットワークは，応答の悪いホストによる影響はそれほど受けないようになっている。さらに，再組立て閉鎖は起きないようになっている。なぜなら，ネットワーク中の複数パケット・メッセージ

に対しては再組立て用記憶域が割付けられているので，目的地ＩＭＰがそのうちの１つのホストへむけられた複数パケット・メッセージを追出すようなことはしないからである。

B．発信地 ― 目的地間のシーケンス制御

フロー制御機構のような基本機能に加えて，もともと，リンク機構も発信地 ― 目的地シーケンス制御の基本をなすものであった。単一リンクでは，１回には１つのメッセージだけしか送れないので，各リンクのメッセージは順序が保たれていた。各リンクには，一連番号をつけることによって重複を検出していた。さらに，ＩＭＰは80ビット以下のメッセージを優先メッセージとして扱い，それが入ってきた時には出力待ち行列中で長いメッセージよりも前の方に入れるという特別な処理をしていた。したがって，短いメッセージはネットワーク中をすばやく通過することができた。

各ＩＭＰはリンク機構に関連した表をもち，この表は大きく，読出しには時間がかかった。リンク機構はフロー制御にとって必要でなくなってしまったので，シーケンス制御にはもっと時間のかからない機構を採用すべきであると考えた。そこで，ＩＭＰサブネットワークからこのリンク機構を取り去ることにした。ＲＦＮＭ信号はリンク上の発信地ホストにまた送り返されるけれども，リンク番号はホストがメッセージを識別するのにしか使用しない。リンクによるシーケンス制御機構を改めるために，各発信地ＩＭＰと目的地ＩＭＰは単一の論理的な“パイプ”で結ばれているという考え方に基づくフロー制御機構をとることに決めた。各ＩＭＰは各パイプに対する独立したメッセージ番号列を保持している。メッセージ番号は，発信地ＩＭＰで各メッセージに割り当て，このメッセージ番号は目的地ＩＭＰでチェックされる。

８ビットのメッセージ番号の空間（このネットワークが完成した時でも十分である）に対して，現在認められている正しいメッセージ番号をチェックするために，発信地と目的地の双方には小さな窓がある。この番号によって，パイプ中を同時にいくつかのメッセージが通れるようになっている。目的地ＩＭＰに到着したメッセージの番号がその窓で指定された範囲を超えている

と，そのメッセージは放棄される。現在のところ，この窓は数字4個分の大きさである。ネットワークに要求される応答時間を考慮すれば，この大きさは大体適切であるといえる。メッセージ番号には2つの目的がある。1つは，パイプ中を通過することができる4つのメッセージの順序づけをすることであり，もう1つは，重複を検出することである。メッセージ番号はＩＭＰサブネットワークに対して内部的に決められたものであり，ホストはわからない。

単一の発信地／受信地パイプに基づくシーケンス制御システムでは，他のトラフィックよりも先行するような優先的トラフィックが許されない。そこで，各発信地と目的地の間に2本のパイプを許すことによってこの問題を解決した。一方を優先（すなわち遅延時間が短い）パイプと呼び，他方を非優先（すなわち高速伝送）パイプという。各パイプがネットワーク中の他のすべてのＩＭＰに対して8ビット・メッセージ番号2個を保持しなければならないというのを避けるために，優先パイプと非優先パイプを合わせてしまい，重複は共通に検出ができるようにした。したがって，各ＩＭＰに対しては，11ビットからなるメッセージ番号列を1個だけもてばよいようになっている。

この11ビットの内訳けは，優先／非優先フラッグに1ビット，優先メッセージの順序を示すのに2ビット，すべてのメッセージに順序を付けるのに8ビットとなっている。たとえば，文字A，B，C，Dが優先メッセージに対する2ビットの順序番号を表し，それらの文字がないものは非優先メッセージを表わすものとすると，つぎのような例が考えられる。発信地ＩＭＰが，非優先メッセージ100，それから優先メッセージ101Aと102B，それから非優先メッセージ103を送るものとする。目的地ＩＭＰは，これらのメッセージを102B，101A，103，100の順序で受け取るものとする。ＩＭＰは，そのホストに101A，102B，100，103の順序でメッセージを渡す。メッセージ番号100が最初に目的地ＩＭＰに到着したとすると，それが最初に目的地ホストに送られてしまうことになる。しかし，優先メッセージはネットワーク中で遅れているので，この優先メッセージはメッセージ番号１００に先行

（ leapfrog ahead ）することが許される。目的地ホストは優先メッセージBよりも前に優先メッセージAを受け取らなければならないので，ＩＭＰは101A が到着するまで102B を保持しておく。同様にして，メッセージ103よりも前にメッセージ100をホストに渡さなければならない。

ホストは，ある１個所の目的地に対するメッセージを同時にいくつか保持するようになることもある。しかし，優先メッセージは他のメッセージに先行して"かえるとび"ができるし，一連の長いメッセージの中で，最後のものが短かかったりすることもあるので，優先順位はメッセージの長さをもとにして正確に割当てることはできない。したがって，メッセージが優先的かどうかは，ホストが明確に表示してやらなければならない。

メッセージの消失という事態が発生したときの処理は，配慮されたメッセージ番号と予約した記憶域を使って，注意深く行なわなければならない。発信地ＩＭＰは，ＲＦＮＭ信号がまだ返ってきていないメッセージのトラフィックを保持している。長時間（現在は約30秒）たってもＲＦＮＭ信号を受けとらなかった時には，発信地ＩＭＰは目的地ＩＭＰに対して制御メッセージを送って，不完全伝送の可能性を問い合わせる。目的地はそのメッセージにこたえて，問い合わせのあったメッセージがすでに受信されているかどうかを知らせてくる。発信地ＩＭＰは応答を得るまでこのような問い合わせをつづける。この技法は，発信地と目的地ＩＭＰがメッセージ番号列の同期を取ることを保証し，機械の障害によるサブネットワーク中でのメッセージの消失という滅多にしか起らないような場合における記憶域の解放を保証している。

C　ＩＭＰ ― ＩＭＰ間の伝送制御

ＩＭＰ－ＩＭＰ間の伝送制御に新しい技法を採用することによって，初期の分離したＡＣＫ信号／タイムアウト／再転送の方式にくらべて効率を10～20％よくすることができた。この新しい方法は非常に遠隔地にあるホストに対しても利用されている。この方法では，各物理的なネットワーク網をたくさんの論理的な"通信路"に分割しているわけで，現在のところ各方向に対

して８つに分割されている。ＡＣＫ信号は通常のネットワーク・トラフィックにのって返される。すべてのパケットは１通信路当たり１ビットからなるＡＣＫ信号ビットの集合を含んでいる。各ＡＣＫ信号もそれ自身パケットにして送るというような初期の方法に比べると，伝送速度に対する要求は以前よりも小さくなる。どれくらい節約されるかについては後述する。そのほか，再伝送の時間が新しいトラフィックの量に依存するようにした。ネットワークの負荷が軽い場合，再伝送の相互干渉が最小になるようにトラフィックを自動的に調整を行なう。

各パケットは出力通信路が割当てられ，その通信路に対する"偶奇"ビット（重複したパケット伝送の検出に用いる），通信路番号，反対の方向の各通信路について１ビットのＡＣＫ信号ビット８個を伝送する。

送信側ＩＭＰは，たえず自分が使用している通信路を走査し（それらの通信路と関連したパケットをもつもの），通信路番号とそれ自身に関連した偶奇ビットによってパケットの伝送を行なう。受信側ＩＭＰにおいて，受信されたパケットの偶奇ビットが正しい受信通信路に関連した偶奇ビットと一致しなければ，そのパケットは受け入れられ，受信偶奇ビットが補なわれる。もしそうでなければ，パケットは重複しており，放棄される。

１つの回線を通って到着したすべてのパケットは，８個の全通信路に対するＡＣＫ信号を含んでいる。これは，伝送される「すべての」パケットの制御部分に入っている８個のＡＣＫ信号に対して確保されている位置に「受信」偶奇ビットをコピーすることによってなされる。他のトラフィックがなかった場合には，"空のパケット"という形でＡＣＫ信号が戻される。空のパケットではＡＣＫ信号ビットにしか意味をもっていない（すなわち，通信路番号と偶奇ビットには意味がない。空のパケットのＡＣＫはない）。ＩＭＰがパケットを受信すると，ＡＣＫ信号ビットと送信用偶奇ビットとを（ビットごとに）比較する。一致したビットがあると，それに対応した通信路は未使用とマークし，対応したパケットは放棄される。そして送信用偶奇ビットが補われる。

多数の通信路と，長距離の回線で起る遅延時間を考えると，複数個のパケットは伝送のため異常に長い時間待機しなければならないという事態の起ることが想定できる。数個の1000 ビット・パケットが伝送されるまで，1文字のパケットが待機するという事態は起したくない。そのようなことをしたら，発信地側からみた実効遅延時間は10倍，あるいはそれ以上になってしまう。つぎに示すような伝送順序付け機構（ transmission ordering scheme ）を設けた。まだ伝送されていない優先パケットは最初に送る。つぎに，まだ伝送されていない正規のパケットを送る。最後に，送り出すべき新しいパケットがなければ，以前に伝送して未承認になっているパケットを送る。もちろん，新しいトラフィックの連続した流れがある時にでも，未承認のパケットは定期的に再伝送される。

ＩＭＰ間の新しい伝送確認手続きをインプリメントするときに，競合の問題が発生した。他にトラフィックがない時に，パケットを連続して再伝送するという方式をとったために，もとのシステムでは起らなかった問題が生じた。もとのシステムでは長時間の時間切れのときにしか再伝送をしなかった。現在，再伝送中のパケットに対してＡＣＫ信号が到着したときには，出力ルーチンは，入力ルーチンがそのパケットを解放しないようにしなければならない。このような予防手段をとっておかないと，パケットを伝送している間に，そのパケット中のヘッダーとデータとが変ってしまうということになる。このような"複合"パケットが回線の他端で受け取られると，決して起りえないような状態が全部起ってしまうという結果になる。このような欠陥を発見するのにひじょうに長い時間がかかった。*

＊同様の問題が発信地 — 目的地フロー制御のレベルでも発生しており，この問題は興味深い。もし，ＩＭＰが隣接するＩＭＰに対して単一あるいは複数パケットの割付け要求を送ったとすると，ＡＣＫ信号を受けるまで，定期的にその要求を再伝送することになる。ＩＭＰがやがて割当て完了メッセージを受けると，ただちに，メッセージの最初のパケットを送り出す。ＩＭＰプログラムでのインプリメンテーションは，最初のパケットと同一のバッファから要求を出すようにした。これは，単にそのパケットの要求ビットをセットしておくだけである。要求が再伝送の過程にある間に割付け完了メッセージが到着したとすれば，最初のパケットとして，再び同じバッファに送る前に，プログラムはその要求が完全に伝送されるまで待機しなければ→

5.4.2　プログラムの構造

ＩＭＰプログラムの基本的な機能は，次に示すような処理を含むパケット処理である。

- ホストからのメッセージのセグメントをパケット構成にすること。
- 蓄積交換用パケットの受信，経路選択および伝送。
- 未確認パケットの再送。
- ホストへ伝送するためにパケットをメッセージに再組立てすること。
- ＲＦＮＭ信号と他の制御用メッセージの生成。

このプログラムは，又，ネットワークの状態を監視したり，統計データを収集したり，オンラインのテストも行なう。

ネットワーク中には25台ものＩＭＰが接続されるようになり，約２年半稼動してみた結果，最近，運用プログラムの大修正を行なった。修正されたプログラムは，前の小節で述べたようなアルゴリズムをインプリメンテーションし，ネットワークの閉鎖を引起す原因を究明し，ＩＭＰのパフォーマンスを向上させた。また，修正されたプログラムは，ＩＭＰの能力を拡張し，その結果，現在では，共用搬送回線を使用してホストとインタフェースがとれるようになり（非常に遠隔地にあるホスト），広範囲にわたる伝送速度をもつ回線に対してもバッファを効率よく管理し，すぐれたネットワーク診断を行なうことができるようになった。

この小節では，最近おこなったプログラムの修正の中で，おもに構造上の変更点について述べる。

Ａ．データ構造

図５－31に従来の，図５－32に現在のコア記憶域の配置を示す。プログラムはそれぞれ機能的に異ったモジュールに分割されていて，各モジュールはコアの１ないし２ページを占めている。ここで，コード*は一般にページ内で

ならない。そのようにしないと，要求ビット，偶奇ビット，ＡＣＫ信号ビットおよびメッセージ番号（複数パケットの要求の場合）が変化してしまう。これがもう１つの問題であった。

＊プログラムの命令部分

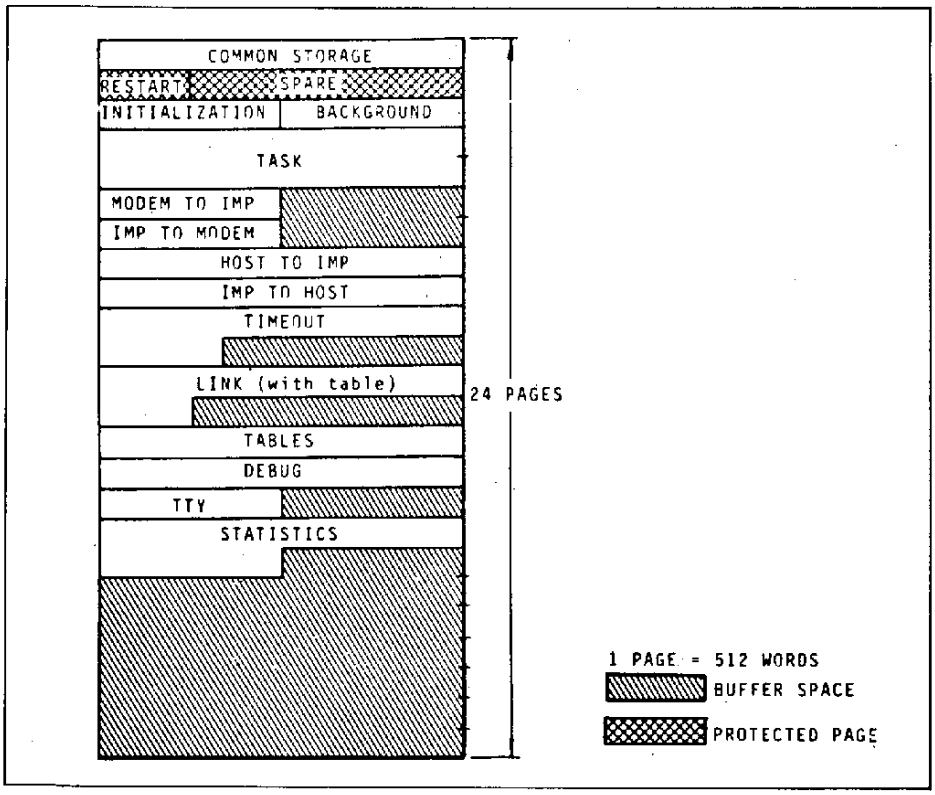


図5－31　Map of core storage(改良前)

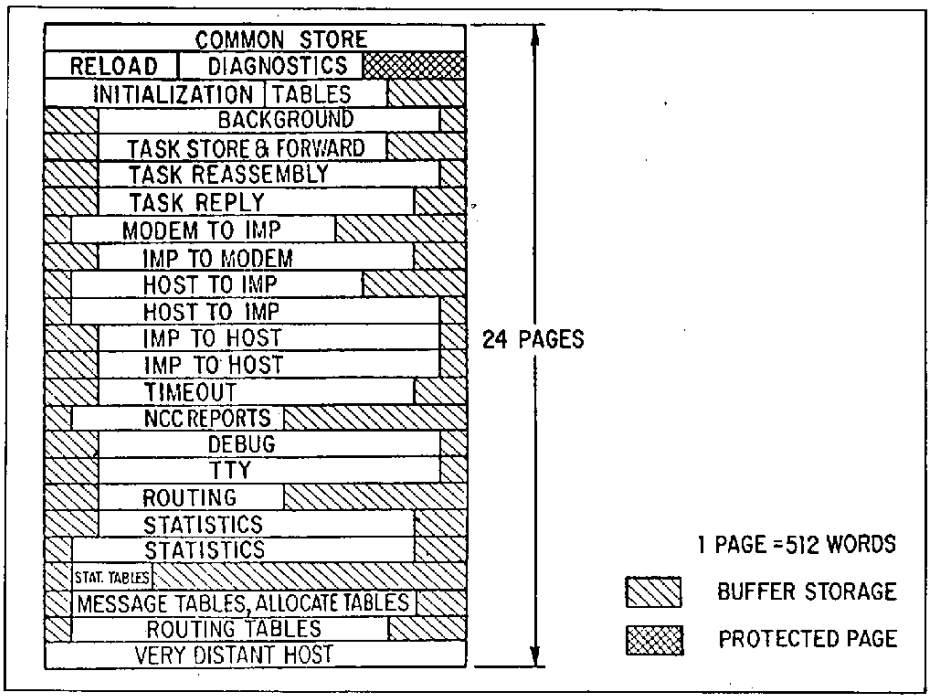


図5－32　Map of Core Storage(改良後)

中央に集めてあり，コアの全ページにコードがあるということに注意してほしい。もとのやり方では，コードはページの先頭から入れるようにし，コードの入ったページは記憶装置の最初の方から使用していくようにしていた（図5－31参照）。したがって，もとの方法だと，記憶域の終りの方に大きな連続したバッファ領域をとることができるが，すべてのページの境界で切れ目ができてしまう。一方，コードを各ページの中央に集めるようにし，あるページにあるコードの最後の語とつぎのページにあるコードの先頭の語との間をバッファとして使用するようにすると，ほとんどすべての切れ目を取り去ることができる。

現在のところ，ＩＭＰには約40個のバッファがあり，ＩＭＰプログラムはバッファを要求してくる各種のタスクに対して，つぎの規則に従い使用可能なバッファを割り付けている。

- 各回線は入出力のためにバッファの共有ができるようになっていなければならない。とくに，各線に対していつでも出力ができることを保証するために，各回線には出力用にかならず１個のバッファが割り付けられている。各回線の入力には，ダブル・バッファリングが適用されており，プログラムによって調べられるすべての入力トラフィックを受け入れられることができるので，ＡＣＫ信号はいつでも処理することが可能になり，バッファを解放する。
- 全部の回線が最大限の能力を発揮できるようにするために，蓄積交換用バッファをできるだけ多く確保するようにする。必要なバッファの個数は，回線の距離と回線の伝送速度に直接依存している。現在，各回線当りのバッファの個数は８以下に制限してあり，全部の回線に対してプールを設けてある。回線の利用率に関する数値的な結果は 5. 4. 3節にいくつか示してある。現在のところ，蓄積交換用バッファのプールでは，最大20個までのバッファが利用できる。
- 再組立て用記憶域としては，10個のバッファが常に確保されており，複数パケット・メッセージと単一パケット・メッセージ２個のための記憶域割

当ては，いつでもできるようになっている。

図５－33にＩＭＰが保持している表の記憶域を要約して示す。すべてのＩＭＰは同一の表をもっている。ＩＭＰプログラムは，このネットワークに現在接続可能な64台の各ＩＭＰについて12語からなる表を保持している。また，ＩＭＰプログラムは，そのＩＭＰに接続されている８台までのホスト（４台は実体のあるもの，４台は仮想のホスト）の各々に対して91語からなる表をもっている。さらに，実際に接続されている各々のホストに対しては，各ホストについて12語のコードを保持している。１台のＩＭＰには５回線まで接続可能であるが，プログラムは，各回線について55語からなる表と37語のコードを保持している。この他に，初期化，統計データ，トレースおよびその他の各種の表をもっている。

初期化を行なうためのコードとそれに関係した表の大きさについては説明しておく必要がある。はじめのうち，これらは非常に小さかった。しかし，ネットワークが拡大され，ＩＭＰの能力が増大するにつれて，初期化のために必要な記憶場所も徐々に大きくなっていった。このことは，ＩＭＰをもはや同一化して扱うことができないという事実による。ＩＭＰは，非常に遠隔地にあるホスト，あるいは，ＴＩＰ（Terminal IMP）ハードウエア，あるいは，５回線と２台のホスト，あるいは，４台のホストと３回線，あるいは，高速伝送回線，さらにやがては衛星リンクといったものを扱わなければならない。ＩＭＰの物理的な組合せが増大し続けていたので，プログラムはすべてのＩＭＰに対して同じものとし，隣接するＩＭＰからプログラムを再ロードすることができ，その他にいくつかの利点を備えているという考えをかたく守った。しかし，プログラムを１つの版しかもたなかったというのは，ＩＭＰの物理的に異なる構成があったときに，それぞれの構成に対してプログラムが正しく働くようにするために，初期化の段階でプログラム自身が自分を再構成しなければならないことを意味する。さらに，ＩＭＰプログラムを隣接するＩＭＰへ再ロードするときには，もとの形に戻せるようになっていなければならない。このような理由から，表とコードを用いるようにした。

ＩＭＰの構成がどれくらい拡張されるものかを予想しなかったために，単純な構成をしたものに対するプログラムとその他の構成に対するプログラムとの差を簡単に計算することはできない。そこで，構成上とくに変った点を陽に表にしておくようにした。

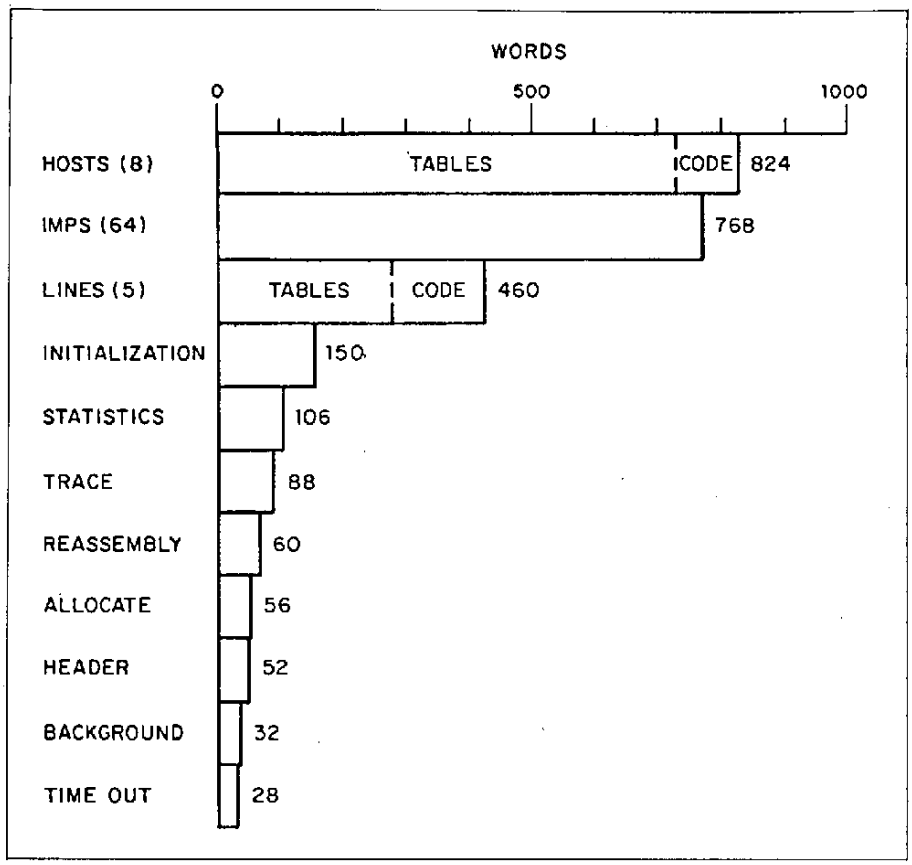


図５－33　Allocation of IMP Table Storage

B．パケット処理ルーチン

パケット・フローとパケット処理を図５－34に示す。ここでは，各種のパケット処理ルーチンの機能を簡単に示し，新しく加わった重要な特徴について説明する。

Host－to－IMP（H→I）　このルーチンは，個々のサイトにあるホストから送られてくるメッセージを処理する。これらのホストは，実際のホス

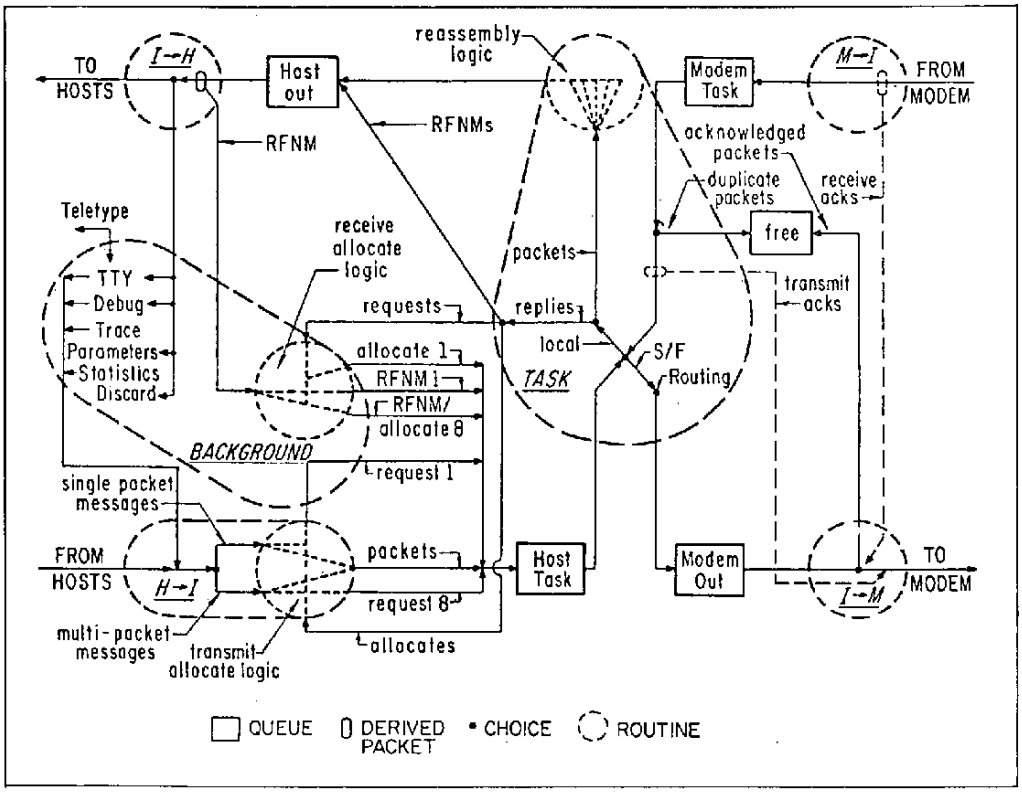


図5－34　Packet Flow and Processing

トでも，仮想のホスト（ TTY[*1]，DEBUG[*2] など）でもよい。このルーチンは，リーダー（ leader ）を使って，メッセージの各々のパケットに前もって付けておかなければならないヘッダーを構成する。又，必要ならば，メッセージに対するリンクの創成，閉塞を行ない，タスク・ルーチンによってさらに処理をしてもらうため，ホスト・タスク待ち行列にメッセージのパケットを入れ，そして，プログラマブル・タスク・インタラプトをトリガする。そこで，このルーチンは，フリー・バッファを獲得し，次の入力の準備をする。

＊1　IMPのTeletypeトラフィックを操作するために使用されるIMPのバックグランド・プログラムの一部。

＊2　IMPのコアー・メモリを検査したり，修正したりするIMPのバックグランド・プログラムの一部。

タスク　このルーチンはパケットをそれぞれの目的地へ正しくふり向ける作業をする。あるローカルなホストへのパケットは，再組立てロジックを通して送り出す。再組立てが完了すると，再組立てが行なわれたメッセージはホスト出力待ち行列を経てIMP－to－Host ルーチンへ渡される。ローカルなＩＭＰへのある種の制御メッセージは，伝送割付けロジック，あるいは受信割付けロジックへ送る。他の目的地へのパケットは，経路選択表指定によってモデム出力待ち行列に入る。

ＩＭＰ－to－Modem（Ｉ→Ｍ）　このルーチンはモデル出力待ち行列からパケットをとり出し順次伝送を行なう。この時，Modem－to－IMP ルーチンが正しく受信し，タスク・ルーチンが受付けたパケットに対するＡＣＫ信号を一緒につけて送り出す。

Modem－to－IMP（Ｍ→Ｉ）　このルーチンはモデムからの入力を処理するもので，正しく受信したパケットをモデム・タスク待ち行列を通してタスク・ルーチンに渡す作業をする。又，このルーチンは，入力されてきたＡＣＫ信号を処理し，正しいＡＣＫ信号が得られたパケットに対するバッファの解放を行なう。

ＩＭＰ－to－Host（Ｉ→Ｈ）　このルーチンは，ローカル・ホストへメッセージを渡し，ＲＦＮＭ信号を発信地ホストに返さなければならないときには，そのことをバックグラウンド・ルーチンに知らせる。

バックグラウンド　このルーチンの機能は，ＩＭＰのコンソール・テレタイプ（ＴＴＹ），デバッキング・プログラム（ＤＥＢＵＧ），統計プログラム（ＳＴＡＴＩＳＴＩＣＳ）[*1]，トレース・プログラム（ＴＲＡＣＥ）[*2] およびコントロール・メッセージを生成するための数個のプログラムからなっている。これらの内で，はじめの４つのプログラムは仮想のホストとして動作する。これらのルーチンは，ホスト／ＩＭＰデータ・チャンネル・ハードウエアのオペレーションをシミュレートするものである。この機能によって，Host

＊1　ＩＭＰの統計を収集するバックグラウンド・プログラムの一部。

＊2　トレースすべきパケットについて収集された情報を伝送するバックグラウンド・プログラムの一部。

－to－IMPおよびIMP－to－Hostルーチンは，実体のあるホストでなくとも，とくにそれが何であるかを知らずに普通に通信をすることができる。このトリックは，命令数を大巾に節約し，ますますそれを使うようになった。まだ伝送されていないメッセージの送信，割付けの送信と返却およびＲＦＮＭ信号の送信を行なうプログラムもまた，このバックグラウンド・プログラム中に入っている。しかし，これらのプログラムは，ホスト／ＩＭＰチャンネル・ハードウエアのシミュレートはせず，仮想のホストとは少し異なった方法で実行される。事実，これらのプログラムはホスト／ＩＭＰルーチンを全く実行せず，むしろ，メッセージをタスク待ち行列に直接入れてしまう。それでもなお，原理は同じである。

タイムアウト　このルーチン（図５－34の中には示していない）は，たくさんの周期的な機能を実行する。これらの機能の内の１つは，ガーベージ・コレクションである。すべての表，ほとんどすべての待ち行列およびプログラムのたくさんの状態をタイム・アウトする。表の中に異常に長い間滞在しているエントリがあったり，異常に長い間ある特定の状態になったままでいるルーチンがあったりすると，このエントリまたは状態をガーベージ・コレクションし，表またはルーチンを初期状態または正常な状態に戻す。このようにして，異常状態によるシステムの停止をほぼ完全に停止する。

表走査をするタイムアウト・ルーチンは，おもしろい方法を使用している。いま，仮に，64個のエントリをもつ表のすべてのエントリを時々，走査しなければならないものとする。その時，タイムアウト・ルーチンはある時間間隔だけ待機しているから，１回で表中のエントリを走査することになる。しかし，そうすると，ＩＭＰプログラムはそのために短時間ではあるがＩＭＰを占有してしまい，全体としてタイミングをとるのがむずかしくなる。そこで，タイムアウト・ルーチンは，１回の実行では１つのエントリしか走査しないようにしてある。実行時間は全体として前者の方法より長くなるが，全体にふりわけられるのでそれほど大きくならない。特に，最悪のタイミングに関する問題（たとえば，１つのモデム入力の終了とつぎの入力開始までの

処理時間）は，このような技法をとり入れることによって大巾に解決された。この技法を利用した顕著な例は，隣接するＩＭＰへの経路選択情報の伝送である。一般的にいって，１台のＩＭＰには５台までのＩＭＰを接続することができる。それ故に，625ミリ秒毎に５台のＩＭＰすべてへ経路選択情報を送ることはせず，上述の技法も用いることによって，125ミリ秒毎に１台のＩＭＰへ経路選択情報を送るようにしてある。

タイムアウト・ルーチンは，プログラムの各種の状態をタイムアウトする以外に，ある時間だけ自分自身をスリープ（ sleep）状態に置いたルーチンを呼起するのに使われる。そういったルーチンの代表的なものは，あるリソースが利用可能になるまで待機するものやタイムアウト・ルーチンのコルーチン（ coroutine ）として作られたものなどがある。タイムアウト・ルーチンによって，それらが再スタートされると，リソースが利用可能になったかどうかをテストし，もしそのリソースがまだ利用できなければ，さらに待機することになる。

5.4.3　パフォーマンス評価

前小節で述べたように，ネットワーク・システムの大規模の変更を行なったので，ＩＭＰのパフォーマンスを再び計算しなおす必要があった。以下の小節では，ＩＭＰのパフォーマンスを再評価したときの結果と，変更前のパフォーマンスとの比較を示す。

A．スループットとメッセージ長

この小節では，ＩＭＰのパフォーマンスに関する２つの尺度，すなわち，最大スループット率と回線トラフィック率を再計算する。スループット率は１秒間にＩＭＰを通過するホスト・データのビット数である。回線トラフィック率はＩＭＰが通信回線を用いて１秒間に伝送できるデータのビット数である。この中には，ＲＦＮＭ信号，パケット・ヘッダー，ＡＣＫ信号，フレーミング文字およびチェックサム文字などのオーバーヘッドが含まれている。

ＩＭＰの最大回線トラフィック率とスループット率を計算するために，ま

ず，１つのメッセージの処理に必要なＩＭＰの計算時間を求める。この計算時間は，１個のメッセージの全パケットとそのＡＣＫ信号およびそのメッセージのＲＦＮＭ信号とそのＡＣＫ信号を処理するのに要する機械命令のサイクルと入出力サイクルの和である。計算時間の計算を簡単化するため，発信地ＩＭＰだけで観測されるホストからメッセージを受信するのに必要な処理時間と，目的地ＩＭＰだけで観測されるホストへのメッセージを送信するのに必要な処理時間は，無視することにする。

パケットは，データがＤビット，ソフトウエアのオーバーヘッドがＳビット，ハードウエアのオーバーヘッドがＨビットである。変更前と変更後のＩＭＰシステムに対するＤ，Ｓ，Ｈの各値はつぎのようになる。

	変　更　前	変　更　後
Ｄ	０－１００８ビット	０－１００８ビット
Ｓ	６４（パケット）＋８０（ＡＣＫ信号）＝１４４ビット	８０ビット（パケット＋ＡＣＫ信号）
Ｈ	７２（パケット）＋７２（ＡＣＫ信号）＝１４４ビット	７２ビット（パケット＋ＡＣＫ信号）

１個のパケットに対する入力／出力処理の時間は，Ｄ＋Ｓビットをメモリから一方のＩＭＰにインタフェースしているモデムに伝送する時間と，Ｄ＋Ｓビットを他方の側にあるＩＭＰのメモリへ伝送する時間の和である。Ｒを１秒間当たりのビット単位の入力／出力伝送率*とすると，１個のパケットに対する入力／出力伝送時間は２（Ｄ＋Ｓ）／Ｒとなる。

したがって，Ｐパケットからなる Ｂ ビットのメッセージに対する全入出力時間 I_m は，２（Ｂ＋Ｐ×Ｓ）／Ｒである。ＲＦＮＭ信号に対する入力／出

＊この計算においては，516 ＩＭＰ（最初使用していたもの）と316 ＩＭＰ（新しいＩＭＰには，すべてこれを使用している）とを区別している。516 ＩＭＰはサイクル時間が0.96 μ secであり，316 ＩＭＰではサイクル時間が1.6 μ sec である。516では４サイクルのデータブレイクをもっていたのに対し，316では２サイクルのデータブレイクを備えている。したがって，入出力転送量は516の場合は384 μ sec 当たり16ビットであり，316の場合は3.2 μ sec 当たり16 ビットである。

力伝送時間I_rは，2S／Rである。

上で求めた3つの入力／出力伝送時間の各々には，プログラムの処理時間Cを加えなければならない。Cの値は，メッセージの1パケットやRFNM信号の処理時間と大体同じである。

もともとのIMPプログラムでは，パケットあたりのプログラム処理時間はつぎのようになっていた。

モデム出力	100サイクル	……パケットの送信
モデム入力	100サイクル	……隣接するIMPでのメッセージ受信
タスク	150サイクル	……それの処理（出力回線の決定）
モデム出力	100サイクル	……ACK信号の送り返し
モデム入力	100サイクル	……最初のIMPでのACK信号の受信
タスク	150サイクル	……ACK信号の処理
	700サイクル	……1パケット当たりのプログラム処理時間

変更したIMPプログラムの場合は，プログラム処理時間は次のとおりである。

モデム出力	150サイクル	……パケット送信とACK信号の送り返し
モデム入力	150サイクル	……パケット受信とACK信号の処理
タスク	250サイクル	……パケットの処理
	550サイクル	……1パケット当たりのプログラム処理時間

最後に，IMPにおける各種の周期的な処理（主として，経路選択計算）に関するオーバーヘッドのパーセントVを追加する。Vは現在のところ約5％である。

以上のことをもとにして，Pパケットからなるメッセージの計算時間L（単位は秒）を求めることができる。

$$L=\underbrace{(P\times C+I_m)\times(1+V)}_{\text{パケット}}+\underbrace{(C+I_r)\times(1+V)}_{\text{RFNM信号}}$$

最大スループット率Tは，単一メッセージのデータ・ビット数をそのメッセ

ージの計算時間で割ったものである。すなわち，

$$T = \frac{L}{B}$$

である。

最大回線トラフィック率R（１秒当たりのビット数）は，

$$R = \text{スループット率} + \frac{\text{メッセージの各パケットとRFNM信号のオーバーヘッド・ビット}}{\text{メッセージの計算時間}}$$

である。すなわち，

$$R = T + \frac{(P+1)\times(S+H)}{L} = \frac{B+(P+1)\times(S+H)}{L}$$

である。

もとのプログラムと変更後のプログラムおよび516ＩＭＰと316ＩＭＰにおいて，メッセージの長さを変えた時の最大スループット率と回線トラフィック率を図５－35に示す。

ＩＭＰシステムのパフォーマンスに関する変化はつぎのように要約できる。

- 蓄積交換用パケットに対するプログラムの処理時間は２０％ だけ減少した。
- 回線のスループット率は，516ＩＭＰの場合では４％，316ＩＭＰの場合では７％だけ増大した。

その結果として，正味のスループット率は，516ＩＭＰで17％，316ＩＭＰで21％増大した。したがって，316ＩＭＰでも，516ＩＭＰを用いてもとのプログラムを処理した場合にくらべ，ほぼ同じ処理をすることができる。516ＩＭＰは，現在約850kbsの処理が可能である。

- 最大長のパケットに対する回線のオーバーヘッドは29％から16％に減少した。

その結果として，電話回線の実効容量は，最大パケット長メッセージが，50kbs の回線で１秒あたり38個処理できたのに対し，このシステムでは43個処理できるようになった。

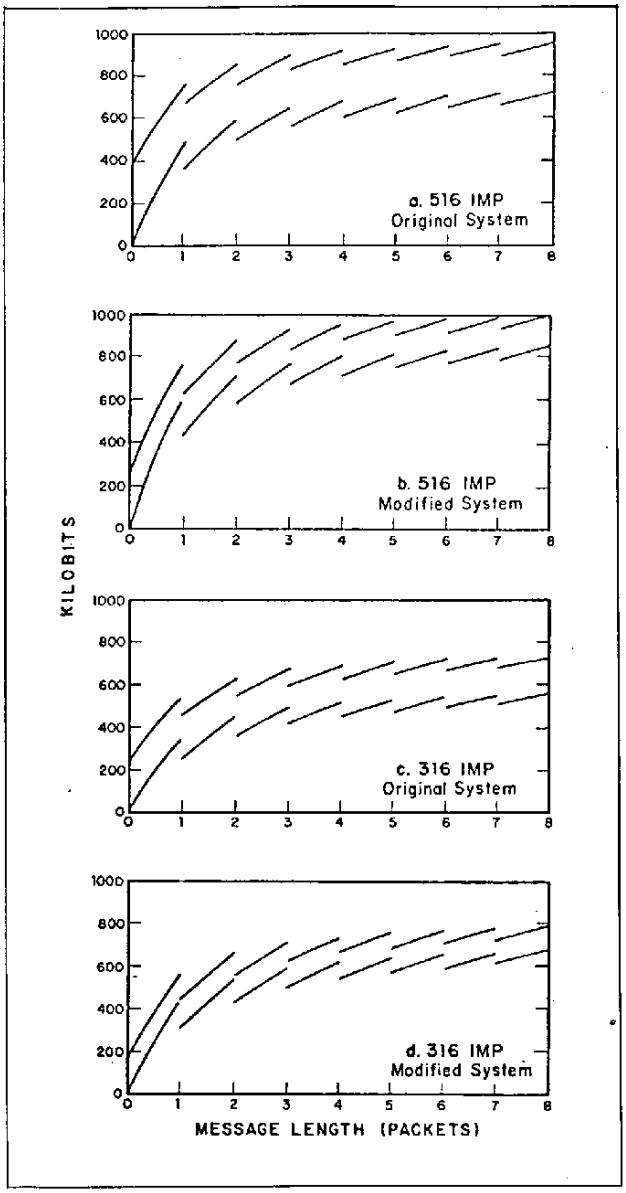


図5－35　Line Traffic and Throughput vs. Message Length. The upper curves plot maximum line traffic, the lower curves plot maximum throughput.

B．周遊遅延時間とメッセージ長

この項では，メッセージの伝送に伴う最小周遊遅延時間を計算する。メッセージはP個のパケットから構成され，H個のＩＭＰを経由して送られるものとする。最初のパケットは，そのパケットの処理時間C，伝送遅延時間T_b

および伝播遅延時間Lによって遅れを生じる。そのあとのメッセージの各パケットは，前のパケットより$C+T_b$だけ遅れる。メッセージのＲＦＮＭ信号は，伝送遅延時間T_rをもつ単一パケット・メッセージなので，全体としての遅延時間は，つぎのようになる。

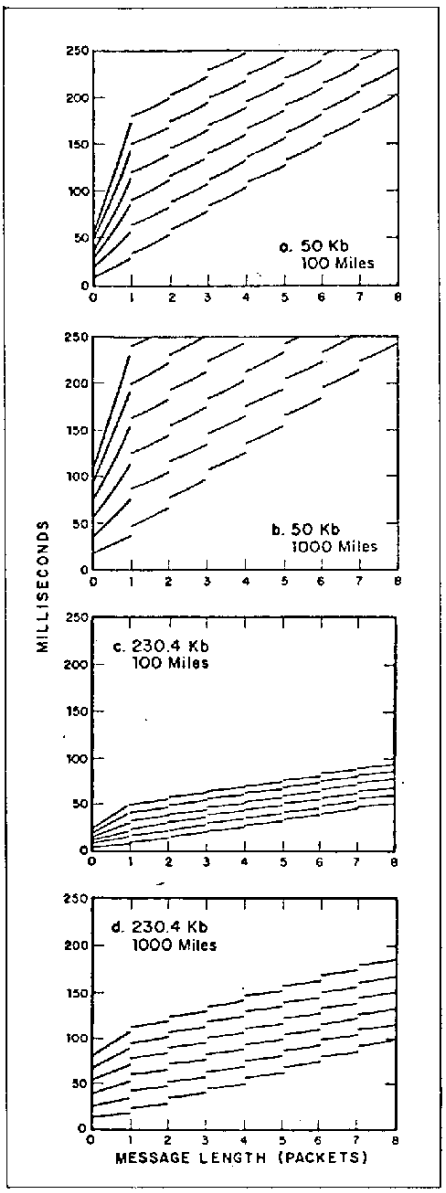


図5－36　Minimum Round Trip Delay vs. Message Length. Curves show delay for 1–6 hops.

$$\underbrace{H\times(C+T_p+L)}_{\text{最初のパケット}}+\underbrace{(P-1)\times(C+T_p)}_{\text{以後のパケット}}+\underbrace{H\times(C+T_p+L)}_{\text{RFNM信号}}$$

単一パケット・メッセージの場合には，この値はつぎのように簡単になる。

$$2H(C+L)+H(T_p+T_r)$$

メッセージ長，途中経由するＩＭＰの数（hop数），回線速度および回線長を変えた時の最小周遊遅延時間を図５－36に示す。これらの曲線は実験データと一致している。

C．回線利用率

通信回線が100％使われた時の負荷に耐えるようにするために必要なバッファの個数は，回線の伝送速度や距離だけの関数ではなく，パケットの長さ，ＩＭＰの遅延時間，伝送確認手続きも含んだ関数である。回線をビジーに保つのに必要なバッファの個数を計算するためには，送信側ＩＭＰがパケットを送り出してから，それに対するＡＣＫ信号を受けとるまでの待ち時間の大きさを知らなければならない。もし回線のエラーがないものとすると，その待ち時間は，パケットとそれに対するＡＣＫ信号の伝播遅延時間P_pとP_A，パケットとそれに対するＡＣＫ信号の伝送遅延時間T_pとT_AおよびＡＣＫ信号が送られてくる前に生じるＩＭＰ処理遅延時間の和である。したがって，回線をフルに利用するのに必要なバッファの個数は，

$$(P_p+T_p+L+P_A+T_A)/T_p$$

である。

$P_p=P_A$であるから，この式は

$$\frac{2P}{T_p}+1+\frac{L+T_A}{T_p}$$

と書き直すことができる。すなわち，回線をフルに使うに必要なバッファの個数は，回線の長さと伝送速度に比例し，パケットの大きさに反比例し，さらに定数項が加わったものである。

T_pを計算するには，短いパケットと長いパケットの混合の具合を考慮し

なければならない。そうすると，T_P はつぎのようになる。

$$T_p = \frac{x T_s + y T_L}{x + y}$$

ただし，x と y は，短いパケットの個数と長いパケットの個数の比である。T_S と T_L は，それぞれ短いパケットと長いパケットに対する伝送遅延時間である。許される最も短いパケットは152ビット長（全部がオーバーヘッド）であり，最も長いパケットは1160 ビット長である。回線の伝送速度が与えられたときに，それに対する T_s と T_L は簡単に計算できる。1.4Mbs の回線を用いたときの T_s は106μ秒，9.6kbs の回線を用いたときの T_L は120.5ミリ秒なので，T_s と T_L は，通常，これらの範囲のある値をとる。

ＩＭＰの処理遅延時間で最悪の場合を想定すると（すなわち，最長パケットの最初のビットが送られてきた時，ちょうどＡＣＫ信号を伝送することができるようになる），

$$L = T_L$$

である。

着信側ＩＭＰでＡＣＫ信号を次に発信側ＩＭＰに伝送されるパケットの中に含ませて返されるのに必要な伝送遅延時間は，多分，平均値*として

$$T_A = \frac{T_s + T_L}{2}$$

となる。

伝播遅延時間Ｐは，本質的には，"光の速さ" の遅延時間である。その値の範囲は，短い回線の場合の50μ秒から，大陸横断回線の場合の50ミリ秒および衛星通信リンクの場合の275ミリ秒までである。

以上のことから，回線の伝送速度，回線の距離とトラフィック・ミクスに対して回線をフルに利用するのに必要なバッファの個数を計算することがで

＊必要なバッファの個数を計算するとき，この仮定のばらつきは，わずか秒の位でしか影響を与えない。

きる。典型的な伝送速度，長さおよびミックスに対する結果が図５－37に示されている。このグラフで注意しなければいけないのは，回線速度が増加するにつれて，曲線の立上がる点がだんだん短い距離の方に移ってくることである。9.6kbs の場合は定数項が大きくきいているのに対し，1.4Mbs の場合になると定数項はほとんで意味をもたなくなっている。また，回線距離に

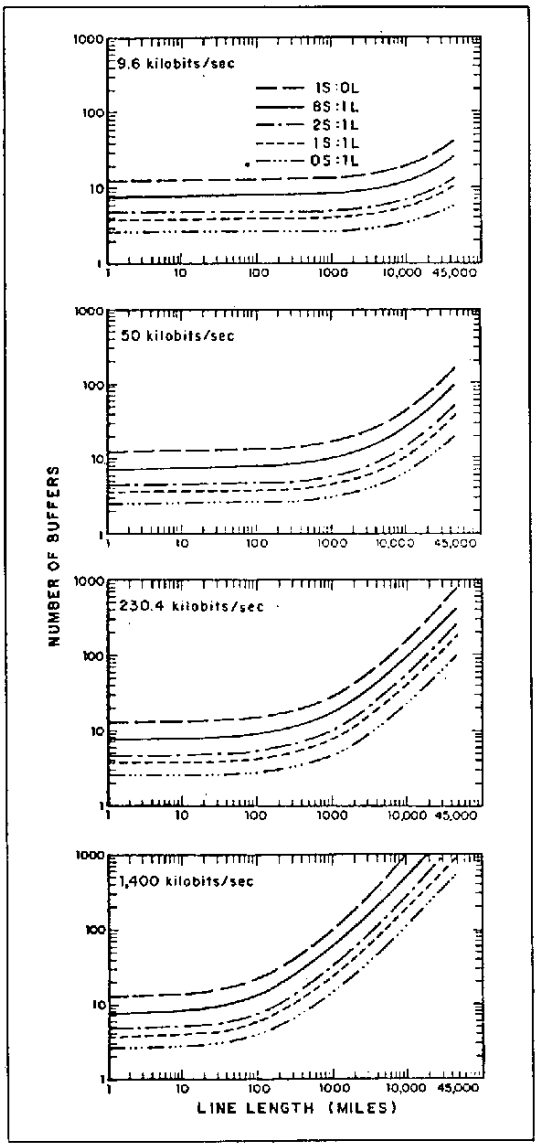


図5－37 Number of Buffers for Full Line Utilization. Traffic mixes are shown as the ratio of numbers of short packets (S) to number of long packets(L).

対して大いに変化することを示すために，各グラフ中のそれぞれの曲線の距離は対数目盛で一定になるようにとってある。

参 考 文 献

1) McKay, D. B. and Karp, D. P. IBM Computer Network/440. Courant Computer Science Symposium 3, December 1970, pp. 27-44.

2) Herzog, B. MERIT Computer Network. Courant Computer Science Symposium 3, December 1970, pp. 46-47.

3) Aupperle, E. MERIT Computer Network: Hardware Considerations. Courant Computer Science Symposium 3, December 1970, pp. 49-64.

4) Cocanower, A. MERIT Computer Network: Software Considerations. Courant Computer Science Symposium 3, December 1970, pp. 66-77.

5) Mendicino, S. F. OCTOPUS: The Lawrence Radiation Laboratory Network. Courant Computer Science Symposium 3, December 1970, pp. 95-109.

6) Farber, D. J. Networks: An Introduction. DATAMATION, April 1972, pp. 36-39.

7) Williams, L. H. A functioning computer network for higher education in North Carolina. Proceedings of AFIPS 1972 Fall Joint Computer Conference, Vol. 41, pp. 899-904.

8) Kirstein, P. T. The future network development in Europe and the USA. International Computer State of the Art Report: Computer Networks Infotech Information, 1971, Vol. 6, pp. 347-363.

9) Price, W. L. Simulation of data transit networks. National Physical Laboratory Com. Sci. 56, April 1972.

10) Flectcher, J. Livemore Time-Sharing System. Lawrence Livemore Laboratory, March 1970, LTSS-1, Edition 2.

11) Heart, F. E., Kahn, R. E., Ornstein, S. M., Crowther, W. R. and Walden, D. C. The interface message processor for the ARPA computer network. Proceedings of AFIPS 1970 Spring Joint Computer Conference, Vol. 36, pp. 551-567.

12) Ornstein, S. M., Heart, F. E., Crowther, W. R., Rising, H. K., Russell, S. B. and Michel, A. The terminal IMP for the ARPA computer network. Proceedings of AFIPS 1972 Spring Joint Computer Conference, Vol. 40, pp. 243-254.

13) Carr, S., Crocker, S. and Cerf, V. Host/host protocol in the ARPA network. Proceedings of AFIPS 1970 Spring Joint Computer Conference, Vol. 36, pp. 589-597.

14) Crocker, S., Heafner, J., Metcalfe, R. and Postel, J. Function-oriented protocols for the ARPA network. Proceedings of AFIPS 1972 Spring Joint Computer Conference, Vol. 40, pp. 271-280.

15) Cole, G. D. Performance measurements on the ARPA computer network. Proceedings of the Second ACM IEEE Symposium on Problems in the Optimization of Data Communications Systems, Palo Alto California, October 1971, pp. 39-45.

16) Frank, H., Frisch, I. T. and Chou, W. Topological Considerations in the design of the ARPA computer network. Proceedings of AFIPS 1970 Spring Joint Computer Conference, Vol. 36, pp. 581-587.

17) Frank, H., Kahn, R. E. and Kleinrock, L. Computer communication network design—Expérience with theory and practice. Proceedings of AFIPS 1972 Spring Joint Computer Conference, Vol. 40, pp. 255-270.

18) Kleinrock, L. Analytic and simulation methods in computer network design. Proceedings of AFIPS 1970 Spring Joint Computer Conference, Vol. 36, pp. 569-579.

19) Thomas, R. and Henderson, D. A. McROSS—A multi-computer programming system. Proceedings of AFIPS 1972 Spring Joint Computer Conference, Vol. 40, pp. 281-294.

20) McQuillan, J. M., Crowther, W. R., Cosell, B. P., Walden D. C. and Heart, F. E. Improvements in the design and performance of the ARPA network. Proceedings of AFIPS 1972 Fall Joint Computer Conference, Vol. 41, pp. 741-755.

21) Kleinrock, L. Performance models and measurements of the ARPA computer network. ONLINE 72 Conference Proceedings, Vol. 2, pp. 61-85.

22) 本間 鶴千代. 待ち行列の理論. 理工学社, 1972

6. 海 外 調 査 報 告

6. 海外調査報告

昭和47年10月22日より約3週間，米英仏3ケ国におけるコンピュータ・システム評価の実状調査をおこなった。

訪問先は大学，研究機関，ユーザなど下記の12個所であった。

大学	Stanford, UCLA
公共研究機関	NBS，NPL(英)，IRIA(仏)
民間研究機関	BBN
ユーザ	保険会社2，銀行1，官庁1(英)
コンピュータ・メーカ	CII(仏)
調査会社	Compata

主な調査目的は次のような点である。

① 大学や研究所におけるシステム評価の研究や今後の傾向に対する意見
　— Stanford，UCLA，NBS，IRIA，Compata

② 商用評価システムのユーザの意見
　— 保険会社，銀行，官庁

③ 政府機関におけるシステム評価に関する活動
　— NBS, NPL, IRIA, 英国政府機関

④ ヨーロッパにおけるシステム評価の現状
　— NPL, IRIA, 英国政府機関, CII

⑤ コンピュータ・ネットワーク・システムの評価
　— UCLA, BBN, NPL

調査結果の概要をコンピュータ・システムとネットワークに分けて以下に報告する。なおより詳しい内容は別冊，当財団資料"コンピュータ・システム評価海外調査報告書"48年1月を参考にされたい。

6.1 コンピュータ・システムの評価

A. 各国におけるシステム評価の実状

今回の調査の目的の一つは国別の比較という観点から見ることであったが，次の５つの項目①政府の積極性，②商用評価システムの開発，③コンピュータ・ユーザの実績，④研究，⑤コンピュータ・メーカの実績，に対して若干比較をおこなってみたい。

① 政府の積極性

米国：国防省を含め米国の連邦政府がコンピュータ・システムの巨大ユーザであることは周知の事実であるが，それだけに新システムの導入の際の機種選択や，既存のシステムの有効利用などの面でシステム効率評価に対し，多大の努力を払って来ている。しかしより重要な点は，民間を指導する立場にある政府として，評価の目安というような何らかの基準を作り，行政指導を行なうか否かということである。日本と米国では政府と民間の力関係がやや異なるとも言われ，少なくとも強制的と思われるような政策はとられていないが，それでも政府自体がビッグ・ユーザである関係上，結果的には類似のニュアンスで受けとられることもあり得る。

例えば，米国商務省の所属研究機関であるＮＢＳがまとめ役として動きつゝあるシステム評価に関する委員会活動はその一つであろう。独自のあるいは商用の評価システムを使用して政府内の導入機種を決定するというプロセスが定着すれば，民間，とくにコンピュータ・メーカへの影響は多大である。そして民間のユーザもこれにならう方向に向うのは当然であろう。

これらを含め，米国における政府の積極性はかなり強いと判断できる。

英国：今回訪問したCivil Service Department の Central Computer Agency は米国のＮＢＳにやや似た性格を持っており，急増

しつつある政府内のコンピュータ・システムやコンピュータ技術者の需要に対して，既存のシステムが実際にどの程度効率よく活かされているかを再検討し，システムの評価という観点から官民を含めた広い範囲のコンピュータ・ユーザに何等かの提言や指針を与える役目を持っているという。このような動きは，ここ１，２年ということであるが，国産コンピュータ・メーカＩＣＬの育成という目的も含めて，日本と事情の似かよった英国の今後の動向は興味がある。

フランス：政府内でも特に軍関係はコンピュータのビッグ・ユーザであり，ＰＴＴ（日本におけるＮＴＴとほぼ似た立場にある）とともにシステム効率評価に対しかなりの実績を持っているといわれるが，政策上の目的は現在のところ少ないようである。

② 商用評価システムの開発

米国：最近の米国における商用評価システムの躍進ぶりはめざましいものがある。COMRESS, Boole & Babbage, Tesdata などの半ば専門のメーカの他に，一般のソフトウエア会社がこの方向に進出を始めている。定評ある製品はそれぞれ数100のオーダのユーザを持ち，ＩＢＭ自身も商用システムを手がけている。

フランス：米国のCOMRESS 社が開発した商用評価シミュレータSCERTと，ほぼ同じ機能を持つPRESTEと呼ぶシステムを国策的なアプリケーション・ソフトウエア会社SEMAが開発し，国産メーカＣＩＩのコンピュータの評価も含め，いくつかのユーザによって使用されている。

英国：今回の調査では国内における商用システムの開発の話題はとくに出なかった。専ら，米国のCOMRESS社，Ｂ＆Ｂ社，Tesdata社などの製品が使われているようである。

③ コンピュータ・ユーザの実績

何処の国においても比較的大きなユーザはそれなりの実績を持つ。特にＩＢＭの機種を使用しているユーザが最も関心が高いのも共通点である。

米国：中級以上のユーザは機種の入れかえ，増設，大型システムの設計等の各段階で，何らかの評価をおこなうのは当然であるというムードである。商用システムもかなり使用されているが，独自の評価システムを開発しているユーザも多い。商用システムとの比は約５０％ づつであるという。

英国：銀行，鉄鋼会社を始め大手ユーザは関心を持ちそれなりの実績もあるが米国にはとても及ばない。

フランス：ＩＢＭやＣＤＣのコンピュータのユーザが中心であるが，軍，電信電話，電力，原子力関係などのユーザはかなり実績を持つ。

④ 研　究

米国：1950 年代から研究が開始され，現在はタイム・シェアリング・システムを始め，より複雑なハードウエア，ソフトウエアを対象にしたもの，または解析的手法の研究がさかんである。

英国：1972 年の９月にシステム評価のコンファレンスが開かれた。モニタリングやシミュレーションが主要なテーマのようであるが，日本でも同年の８月に最初のシステム評価シンポジウムが開かれ，その点では似たような状態である。

フランス：フランスはもともとシミュレーションのさかんな国であるが研究テーマとしてはタイム・シェアリングやマルチ・システムのシミュレーションが主体である。モニタリングの研究は今後の問題である。

⑤ コンピュータ・メーカの実績

米国：新しいシステムを開発する場合，あるいは既製のシステムを改良する目的でシミュレーションを始め何等かのモニタリングをおこない製品に反映させるということはメーカとしてむしろ当然であるが，これらの自分のための評価から除々にユーザの為のサポート，即ちＯＳの一部に何等かの評価機能を内蔵するという動きが加えられつつある。ＩＢＭのＳＭＦはその一つの例であろう。もっともこの傾向はメーカ側の自発的意志というより，ユーザ側からの強い要請が反応したのだ

という見方もある。

英国：ＩＣＬの国産コンピュータではＯＳに何等かのモニタリング機能を内蔵するという様なことはまだ考えていないようである。もっぱら自社のためのシミュレーション・ワークが主体である。

フランス：英国同様，モニタリングに関してはまだあまり関心はない。ハードウエアを中心としたシミュレーションが主なワークのようである。

以上のような比較項目を日本の状態に対応させてみると⑤のコンピュータ・メーカの実績がやや優位であるだけで他は英，仏と似た様な程度であろう。①～⑤のいづれの項目に対しても米国にはとても及ばない。

B. 商用評価システムの使用について

今回の調査の範囲では，よく使用されているシステムは，シミュレータではSCERTとCASE，ハードウエア・モニタではDYNAPROBE，ソフトウエア・モニタではSMS，そして言語プロセッサの評価システムでは，COBOL Optimizerの評判がよかった。この他シミュレータのＣＳＳがＩＢＭの強みで最近はユーザが増加しつつある様子で，同様に新しいシミュレータＳＡＭも期待されているという。

主な意見としては次のようなものがある。

(1) シミュレータ

○SCERTについて

① 高価である。

② 小型でシンプルなシステムに対しては大へん精度がよい。

③ パラレル・プロセッサを持つ機械（例CDC6600）や複雑なシステムには無理である。

④ オンライン，ＴＳＳなどには不向きである。

⑤ 1971 年頃 370の結果は精度が悪かった。

⑥ モデルが明確でないので結果がやや不安である。

今回の調査のうち，商用評価システムの４つのユーザはすべてSCERTの使用者または使用経験者であった。そのうち３ユーザは２，３年または

4，5年前に使用を終了しているが，止めた理由としては高価であるという理由の他，もうシミュレーションの時期ではなくモニタリングの時期であるというユーザが2社あった。SCERTの主な使用は機種選定，増設機器による効率の予測，データ・フォーマットの設計等であるが，一応システムが落着くと，モニタリングに切りかえて，より細かい評価をするというのが一般的のようである。

○CASEについて

① SCERTにくらべると大へん安い。

② SCERTより制約が少なく使い易い。

③ 解析モデルも含まれている。

④ モデルが比較的明確である。

CASEユーザは2社であったが，いづれも旧SCERTのユーザであった。Tesdata社はCASEシステムに関するかなりくわしい資料をユーザに提供するというのが一つの利点でもあるらしい。

○SAMについて

これはまだ新しいシステムで実際の使用者はいなかったがNBSは，有望視していた。

これらシミュレータ全般に言えることはモデルがある程度クリアーであるか，またはユーザ自身が多少モデル記述をおこなうタイプのものは，比較的結果の判断がし易いが，さもないと結果に対する不安がある。そのせいか，ある程度進んだユーザ，あるいはオンライン，TSSなどの評価を試みるユーザは，これらの商用システムの使用をあきらめGPSS，SIMSCRIPTあるいはFORTRAN，PL／Iなどで自分でモデルを作成し，独自のシミュレーションを行っているようである。

商用シミュレータの殆んどは，結果の精度は約5％であると公表しているが，ユーザからはむしろ10％に近いという意見もあった。またシミュレーションは導入前も含めたどちらかというと初期の段階で使用し，一応システムが落着くとモニタリングに切りかえて，より細かい評価をする

というのが一般的のようである。

(2) ソフトウエア・モニタ

○ＳＭＳについて

① 大へんよいシステムであり頻繁に使っている。

② ＰＰＥはＣＯＢＯＬに有力である。

③ 結果の解析にやや技術が必要である。

訪問したユーザの殆んどが期せずしてＩＢＭのユーザであったためこのＳＭＳは全般的に使われていた。一般プログラマーには特にＰＰＥ(Problem Program Evaluator)が多く利用されているようであった。

○ＯＳ３６０のＳＭＦについて

① オーバーヘッドが少なくてよい。

② やや制約がある。例えばＣＰＵタイムの累積をタスク単位でとることができない。

③ 自分で後仕末をせねばならない。

○COBOL optimizerについて

現在使用中，または使用を計画中のユーザが２社あった。大変有効であると評判がいい。このシステムはＣＯＢＯＬのプログラムを入力するとオプティマイズされたオブジェクト・プログラムを出力するものであるが，オプティマイズされたソース・プログラムを出力するSTAGEⅡ(Tesdata社)というシステムもある。

ソフトウエア・モニターはマシン・ディペンデントになる傾向があるので，ＩＢＭユーザの利用度が圧倒的である。米国における商用のソフトウエア・モニターの約80%はＩＢＭの機種用のものであるという事情からも当然と言えよう。

ソフトウエア・モニタの使用はある程度の技術レベルが要求される。特に結果の評価にはそれなりの眼が必要である。逆に言えば，技術レベルの低いユーザは，充分に使いこなせない。モニター・システムにはマルチ処理やマルチアクセスのシステム全体の総合的な効率を測定するものと，

ある言語プロセッサーの効率，あるいは，その言語で組んだユーザ・プログラムの効率などを測定するものとがあるが，前者はコンピュータの管理をする特別な技術者が使用し，後者の言語に関するものは個々のプログラマーが各自で使用するという形が多いようである。

(3) ハードウエア・モニタ

現在ハードウエア・モニタを使用していたユーザは比較的少なかったがDYNAPROBEやX-rayなどを使いたいという意向はかなりあった。現在使用していない理由としては，

① 使用するのに専門的技術，あるいはコンピュータ・メーカの協力が必要である。

② 高価である。

③ 最近のコンピュータ（例えば370）にはコンソールに簡単なモニタリング機能がついているので必要がない。

などの意見があった。一方DYNAPROBEのユーザは，きわめて有効なトゥール（tool）であると言っていたが，十分使いこなすのはむずかしい様子であった。

ハードウエア・モニタとソフトウエア・モニタは両者の共通点もあるが，それぞれの特長を持っており，両方を併用するのが望ましいという意見が多い。

(4) 全般的意見

① 商用評価システムの使用による効率の向上は，システム全体としては20～30％，1つのプログラムでは50％近い向上が経験されたものもあるという。言語ではCOBOLに特に効果があるようである。

② これらのシステムにより指摘された効率上のポイントは

- ○ＣＰＵの利用度（Utilization）
- ○各チャンネルの利用度とバランス
- ○ディスク上のファイルの配置
- ○ファイルのブロッキング・ファクターやレコード・サイズ

○オーバレイ（overlay）構造

○モジュール化の設計

○データの型の定義

○各種のキューの大きさやスケジュール・アルゴリズム

等であるという。

③ コスト・バランスについて

これらの商用システム自身の価格と，それから得られるゲイン（gain）とのコスト・バランスにつき各ユーザに共通の質問をおこなったが，4ユーザのうち，定量的な数字を示してくれたのは1ユーザだけでそれによると年間50,000～60,000ドルのレンタル費に対し，約200,000ドルの節約が可能だということであった。約40％である。

あるユーザからはコンピュータ・システムのコストに比べればこれらのシステムの費用は1～2％位であるから殆んど気にしないという返事もあった。

しかし，このコスト・バランスの評価はかなりむづかしい問題であり，あるユーザはその商用評価システムのコスト効率自身を評価するのも一つの目的であるという。

(5) 第三者の意見

大学，研究機関などで聞いた，これら商用システムに対する意見としては次のようなものがある。

① 商用評価システムの中にはかなり有効なものもあるが，概して対象とするシステムがシンプルなものであるという限定があり，より新しい複雑なシステムに対しては不充分である。

② 一般に高価で，汎用システム故の欠陥もあり，進んだユーザは自主開発をする傾向がある。しかしそのためには何等かの商用システムの使用経験があることが望ましく，その意味では商用システムはトレーニング用に非常によい。

③ 商用システムはすぐ目に見える経済的効果を上げ得るということで有

効である。

④ 評価システム自身の評価が必要である。一般にシミュレーションの結果の精度は実測によって確かめることが望ましい。

C. 総合的な意見

研究機関，ユーザを問わずシステム評価に関する一般的意見としては次のようなものがあった。

① システム評価の技術としてはモニタリング，シミュレーション，アナリティック手法などがあるが，これらはお互いに相おぎない合って全体のレベルが上ることが望ましい。例えばアナリティック手法はシミュレーション・モデルのシステム・パラメータを探すために役立つが，実は最後の目標は矢張りアナリティック・モデル自身のパラメータの研究である。

また，シミュレーションのモデルの正当性のチェックのためにモニタリングは非常によいトゥールである。

② システム評価のための特殊のシミュレーション言語の必要性は，それほどは強くない。たゞ汎用シミュレータは概して処理時間がかかるので，専用のシステムで効率をあげるということは必要かも知れない。しかし，そのための最適のシミュレーション言語あるいはシステムを設計するのはかなり大へんなことであるし，そのインプリメントも決して簡単ではない。むしろ既製の汎用言語に何等かの機能をつけ加える方がよいのではないか。例えば現在の汎用言語ＧＰＳＳに既製のモデルを結合する機能をつけ加えると便利である。

③ シミュレーションの結果の誤差は機種選択の目的には５％以内，システムの増強や設計のためには10～20％位でもよい。

④ ハードウエア・モニタとソフトウエア・モニタは得意の領域がやや異なるので，併用するか，あるいは両方の機能を混合した装置が望ましい。

⑤ 一般にモニタリング・システムは入力の仕方がよろしくない。また，結果の分析ももっと自動的になるべきである。

⑥ 従来のコンピュータは速度とか容量とか，その豊富な機能や大きさが主

として注目の対象になっていたが今後コンピュータ・メーカは，ハードウエアもソフトウエアも効率評価を前提としてシステムを設計するようになるであろうし，なるべきである。

⑦ コンピュータ・メーカはモニタリングや評価機能をシステムにもっと組み込むべきである。ユーザレベルで挿入するよりも誤差やオーバヘッドが少なく，よりきめの細かいモニターをおこなうには，ＯＳ設計時に組み込むのが最もよい。

⑧ 今後，数多くの評価情報や資料が世の中にゆきわたり，ユーザは評価に対するデシジョンがし易くなるであろう。それだけにメーカは苦しくなる。

D. システム評価に関する総合的な感想

ＮＢＳでは特殊なハードウエアをＴＳＳの端末とセントラル・コンピュータの間に挿入し，ＴＳＳ端末ユーザの振舞（Behavior）を測定し，人間と機械を総合したシステムの評価をこころざしている。一方BOSTONのある銀行ではオンライン端末のオペレータが単位時間内に幾つのメッセージの処理が出来るかを測定し，評価をおこなっている。

これらの話はどちらも共通点があり，人間も含めて一つのシステムと考え，総合的に評価すべきであるという方向に向って来たことが感じられる。それにしてもこういった気運の前提としては，少なくとも米国においては端末からのオンライン使用が全く一般化したという証拠でもあろう。

商用評価システムの普及は他国にくらべ米国が群を抜いている。そしてまたこれらの商用評価システム・ユーザは少なくとも２つ以上のシステムを併用するのが一般的であるらしい。シミュレーションとモニタリングはたしかに異なった目的を持っており，一つのシステムを使用してみると，連想的に他のシステムを使用したい要求や必要性が出てくるもののようである。

商用であるなしにかかわらず評価システム自身を評価する気運が出て来ているのは当然であろう。また米国はもとより，英国においても政府機関が特殊な立場で評価問題をとり上げているのは注目される。

ますます複雑化し，且つ絶え間なく変貌を続けるコンピュータ・システム

を，正しく評価するということは非常にむづかしい問題であり，より新しい多角的，総合的な評価技術の研究が今後益々必要となって来るであろう。

また，特にメーカにとっては比較評価のデータが直接営業上の利害関係に影響するだけに（現実に米国では一部でこの問題が起っている）純技術的問題だけは片付かぬ深刻な面も持ち合わせていることを痛感せざるを得ない。

6.2 コンピュータ・ネットワーク・システム

今回の調査でかなり強く印象づけられたのはコンピュータ・ネットワークの計画が予想以上に処々方々で持ち出されているということである。まさにネットワークの花ざかりの感がある。

今回の調査で話題になった計画中のものだけでも次の5つがある。

① 英国のNational Network

Englandの南半分とWalseの1部を含み，18のノードと31のリンクを持ち1.5MBPSの高速回線で結ぶ計画である。

② フランスのCYCLADESネットワーク

Paris郊外のIRIA，CIIなどを中心にGrenoble, Toulouse, Rennes地方の大学，研究機関を結ぶもので最高2MBPSの高速ネットワークである。

③ Europian Information Network

英国，フランス，西独，スイスの5つの主要研究機関をノードとした国際的なネットワークである。

④ 米国のNational Science Fundation Computer Network

⑤ Insurance System of America（ISA）のネットワーク

米国の保険会社を結ぶネットワークである。

これらの殆んどは異質のコンピュータをつなぐリソース・シェアリング（resource sharing）のARPAタイプのものであり，少なくともARPAの一応の成功が強い刺激となったに相違ない。

今回の調査は，これらのネットワークの計画自体を調査することが目的ではなく，ARPA，NPL 等既存のネットワーク・システムの使用経験を通しての評価という立場が主体であったが，このようなネットワークの使用経験自身がまだきわめて限られており，コンピュータ・システムの評価と同じレベルで考えることは無理な状態である。しかしARPANETにおけるメジャメント（measurement）の研究および実績はさすがであり，事前のシミュレーショ

ンや解析的予測データとの対比をはじめ後続のネットワーク計画に多大の貢献をおこなっている。巨額な資金を投じたこのプロジェクトに対して，世間の眼は決して好意的なもののみではないようであるが，一見この無鉄砲とも思われるアイデイアをともかくも実現した事の功績は見事である。

ＡＲＰＡＮＥＴは現在約30のサイト（ site ）がリンクされており，そのうち約３分の１がＴＩＰ（ Terminal IMP ）である。このネットワークはまだ一般には公開しておらず，ＡＲＰＡプロジェクトに何等かのコンタクトを持つ人のみが利用している状態であるが，その範囲内の使用量は除々に増加している。しかし，そろそろ一般に公開すべきだという声もないわけではなく，特にＴＩＰの利用の要求が多いという。２，３年後には何等かの民間団体によって商用ベースで運用される可能性もあるとのことであったが，料金制度やサービス制などに関してはかなり問題がありそうである。

現在の使用法は

- データ転送
- バッチ処理
- インタラクティブ処理

の順であり，バッチ処理ではUCLAのＩBM360／91やSanta Barbara のＩBM 360／75 などの大型機が多く利用されているという。ＩLLIAC－Ⅳ はまだ連結されていない。

大きなデータ・ベースの共用というアプリケーションはまだ極めて限られているが，天気情報，文献情報等の数種類のデータ・ベースとその検索システムが，現在開発されつゝある。

ネットワーク自体はかなり安定しているが，HOST側のトラブルが安定性および効率に影響しているという。

今後，ＴＩＰの増設も含めてサイトの数も増加する一方，より高速（230KBPS，1.3MBPS など）のＩＭＰの研究もおこなわれており，ヨーロッパ，ハワイ等への延長も含めて，このネットワークは更に機能の拡張が計画されている。

ARPAにくらべれば規模は少さいがNPLがその独得なパケット・スイッチング（Packet switching）技術を実現し，研究所内のネットワークを実用に供しているのは立派である。

将来はNational Networkのサブネットとなる可能性もあるが，NPL自身は，この英国のNational Networkの技術的サポートをおこなう中心的立場にあり，将来計画の一環として，実験研究の目的もかねて，研究所内のネットワークをまず実現したということである。

ネットワーク・システムの効率のポイントは一般のコンピュータ・システムとほぼ同様に矢張りスループット（throughput）とメッセージのディレイ・タイム（delay time）であろう。平均的なスループットを大きくし，且つディレイ・タイムを最少にするために，ネットワークの構成，ルーティング（routing），フロー・コントロール（flow control）等の最適化が必要となる。またコンジェスション（congestion）の予防も重要な点である。勿論メッセージ長のきめ方やコード変換の手法，各種インタフェースの取り扱い，その他のソフトウエア上の規約やアルゴリズムも重要な役割りを果すが，さしあたりはメッセージ・スイッチング用のコンピュータ（ARPAの場合はIMP）のハードウエア機能が最も基本的に効率に影響する。UCLAのProfessor KleinrockによればARPAの場合IMPのメモリー容量の制約が全体に非常に大きな影響を与えているという。

一方，見方を変えると如何に効率よくリソースがシェアリング出来るかという事や，どのようなソフトウエア・リソースやデータ・ベースが利用できるかという実用上の評価もある。

しかし，何といっても最大の難関はコストの問題であり，ARPA を始め現在計画中のものもすべてこのコスト問題に直面していると言ってよい。本来，小型，中型コンピュータの氾らんや重複をさけるためにネットワークを建設して効率よくリソースをシェアリングすべく計画されたケースが多いのであろうが，少々の重複がおこったとしても，ネットワークの建設自体よりは経済的負担は少なくてすむという意見も多いそうである。何を基準にコスト評価をする

か，またどの位将来を見越して評価するかが一つの鍵であろう。

この経済的問題は何処でも話題になったが，しかし一方，何人かの発言にあったように各国のNational Networkの計画が除々に進み，更にそれぞれをサブネット（ subnet ）と見なして，サテライト中継も含めてお互いにNational Network同志をリンクし，将来は殆んどの国がネットワークを通してリソースをシェアするという構想が果たして単なる夢物語で終るのかどうか，遅ればせながら日本もその可能性を試みる実験の一端をになう時期にあるのかも知れない。

昭和48年3月発行

発行所　財団法人　日本情報処理開発センター
東京都港区芝公園3丁目5番8号
機械振興会館内
TEL（434）8211（代表）

印刷所　三協印刷株式会社
東京都渋谷区渋谷3－11－11
TEL（407）7316

47－R004